软件开发自学视频教程



配套自学视频上百集
编程实例分析上千个
典型模块分析近20类
项目案例分析近20套
《程序员求职面试宝典》电子书1部

Android自学视频教程

（Android开发）

69集配套视频＋2400项配套资源

- **编程实例分析**1340个
- **典型模块分析**17类
- **项目案例分析**17套
- **数学及逻辑思维、IT英语能力测试**616项
- **实践训练任务**99项
- **《程序员求职面试宝典》（面试真题、技巧及职业规划）**1部

软件开发技术联盟◎编著

◎配套自学视频上百集

几乎覆盖全书所有实例，先听视频讲解，再仿照书中实例实践，会大幅提高学习效率。

◎编程实例分析上千个

各类实例一应俱全，无论学习哪一章节，都可从中找到相关实例加以练习，相信对深入学习极有帮助。

◎典型模块分析近20类

既可作为综合应用实例学习，又可将模块移植到相关应用中，从而避免重复劳动，提高工作效率。

◎项目案例分析近20套

从需求分析、系统设计、模块分解到代码实现，几乎展现了项目开发的整个过程。

◎《程序员求职面试宝典》电子书1部

各类面试真题、面试技巧、程序员职业生涯、简历设计、IT企业中的自身修养等帮助读者更好就业和长远发展。

清华大学出版社

软件开发自学视频教程

Android 自学视频教程

软件开发技术联盟　编著

清华大学出版社

北　京

内 容 简 介

《Android 自学视频教程》以初学者为主要对象，全面介绍 Android 应用开发相关的各种技术。内容编排由浅入深，结合丰富的图解和形象的比喻讲解，并附有大量的注意、说明、技巧等栏目，夯实读者理论技术，丰富管理与开发经验。

《Android 自学视频教程》分 3 篇共 21 章，其中，第 1 篇为入门篇，主要包括 Android 入门、搭建 Android 开发环境、认识 Android 模拟器、剖析 Android 程序、Android 常用组件的使用、掌握布局管理器、Android 程序调试与错误处理、Activity 的使用和使用 Intent 进行通信等内容；第 2 篇为提高篇，主要包括 Android 高级组件的使用、Android 中的事件处理、数据存储技术、Content Provider 实现数据共享、图形图像处理技术、利用 OpenGL 实现 3D 图形、多媒体应用开发、线程与消息处理、网络编程技术和 Service 服务的使用等内容；第 3 篇为实战篇，主要包括 Android 游戏——数独游戏和 Android 应用——家庭理财通两个实战项目。另外本书光盘含：

21 小时视频讲解/1340 个编程实例/17 个经典模块分析/17 个项目开发案例/99 个编程实践任务/616 个能力测试题目（基础能力测试、数学及逻辑思维能力测试、面试能力测试、编程英语能力测试）/23 个 IT 励志故事。

本书适用于 Android 应用开发的爱好者、初学者和中级开发人员，也可作为大中专院校和培训机构的教材。

图书在版编目（CIP）数据

Android 自学视频教程/软件开发技术联盟编著．—北京：清华大学出版社，2014
软件开发自学视频教程
ISBN 978-7-302-37112-0

I．①A…　II．①软…　III．①移动终端-应用程序-程序设计-教材　IV．①TN929.53

中国版本图书馆 CIP 数据核字（2014）第 146027 号

责任编辑：赵洛育
封面设计：李志伟
版式设计：文森时代
责任校对：马子杰
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：203mm×260mm　**印　　张：**35.5　**字　　数：**929 千字
（附 DVD1 张）
版　　次：2014 年 12 月第 1 版　**印　　次：**2014 年 12 月第 1 次印刷
印　　数：1～4000
定　　价：79.80 元

产品编号：051620-01

前　言

Preface

本书编写背景

为什么一方面很多毕业生不太容易找到工作，另一方面很多企业却招不到合适的人才？为什么很多学生学习很刻苦，临毕业了却感到自己似乎什么都不会？为什么很多学生到企业之后，发现很多所学的知识用不上？……高校课程设置与企业应用严重脱节，高校所学知识得不到很好的实践，本来是为了实际应用而学习却变成了应付考试，是造成如上所述现象的主要原因。

为了能满足社会需要，有些人不得不花费巨额费用、花费半年到一年时间到社会再培训，浪费了巨大的人力物力。有没有一种办法让学生在校就能学到企业应用的内容呢？——本书就是为此目的而来。本书从没有编程基础或稍有编程基础的读者层次开始，通过适合自学的方式，从基础知识到小型实例到综合实例到项目案例，让学生在学校就能学到企业应用的内容，从而实现从学校所学到企业应用的重大跨越，架起从学校通向社会的桥梁。

本书特点

1．从基础到项目实战，快速铺就就业之路

全书体例为：基础知识+小型实例+综合实例+项目实战，既符合循序渐进的学习规律，也力求贴近项目实战等实际应用。基础知识是必备内容；小型实例则通过实例巩固基础知识；综合实例则是在进一步综合应用基础知识的前提下，通过模块的形式让内容更加贴近实际应用；项目实战则是展现项目开发的全过程，让读者对基本的项目开发有一个全面的认识。

2．全程配套视频讲解，让老师手把手教您

本书配书光盘含配套视频讲解，基本覆盖全书内容，学习之前，先看、听视频讲解，然后对照书模仿练习，相信会快速提高学习效率。

3．配套资源极为丰富，各类实例一应俱全

（1）**实例资源库**：包括**上千个编程实例，各种类型一应俱全**，无论学习这本书的哪一章节，都可以从中找到相关的多种实例加以实践，相信对深入学习极有帮助。

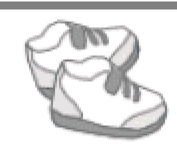
（2）**模块资源库**：包括了最常用的**十多个经典模块分析**，它们既可作为**综合应用实例学习，又可移植到相关应用中**，进而避免重复劳动，提高工作效率。

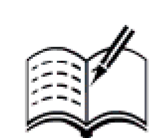
（3）**项目（案例）资源库**：包括**十多个项目开发案例，从需求分析、系统设计、模块分析到代码实现**，几乎全程展现了项目开发的整个过程。

（4）**任务（训练）资源库**：共计**千余个实践任务**，读者可以自行实践练习，还可以到对应的网站上寻找答案。

（5）**能力测试资源库**：列举了**几百个能力测试题目，包括编程基础能力测试、数学及逻辑思维能力测试、面试能力测试、编程英语能力测试**，便于读者自我测试。

（6）**编程人生**：精选了**二十多个 IT 励志故事**，希望读者朋友从这些 IT 成功人士的经历中**汲取精神力量**，让这些经历成为您不断进取、勇攀高峰的强大精神动力。





Note

如何高效使用本书

建议首先看相关实例视频，然后对照图书的实例，动手操作或者运行程序，反复体会，之后再打开本书光盘的“自主学习系统”，找一些对应的实例练习。当然，还可以参考“自主学习系统”的其他资源，加以补充和拓展。

本书常见问题

1．编程软件的获取

按照本书上的实例进行操作练习，需要事先在电脑上安装相关的语言或工具的开发环境（编程软件）。本书光盘只提供了教学视频、自主学习系统等辅助资料，并未提供编程软件，读者朋友需要在网上搜索下载，或者到当地电脑城、软件经销商处购买。

2．关于本书的技术问题或有关本书信息的发布

（1）读者朋友遇到有关本书的技术问题，建议先登录：www.rjkflm.com，搜索到本书后，查看该书的留言是否已经对您的相关问题进行了回复，以避免浪费您更多的时间。

（2）如果留言没有相关问题，可加入 QQ：4006751066 咨询有关本书的技术问题。

（3）本书经过多次审校，仍然可能有极少数错误，欢迎读者朋友批评指正，请给我们留言，我们也将对提出问题和建议的读者予以奖励。另外，有关本书的勘误，我们会在 www.rjkflm.com 网站上公布。

3．关于本书光盘的使用

本书光盘只能在电脑光驱（DVD 格式）中使用，光盘中的视频文件双击即可自行播放。极个别光盘视频文件如果不能打开，请暂时关闭一下杀毒软件再打开；若仍然无法打开，建议换台电脑后将光盘内容复制过来后打开（极个别光驱与光盘不兼容导致无法读取的现象是有的）。另外，盘面若有胶水等脏物建议先行擦拭干净。

关于作者

本书由软件开发技术联盟组织编写。该联盟由一家有十多年集软件开发、数字教育、图书出版为一体的高科技公司——明日科技和一些中青年骨干教师组成。

本书主要由王小科、王国辉执笔编写，其他参与本书编写的人员有张鑫、杨丽、陈英、高春艳、赛奎春、刘佳、辛洪郁、崔佳音、周佳星、刘丽艳、刘红艳、高飞、郭铁、王敬杰、张金辉、刘志铭、宋晶、宋禹蒙、王雨竹、张彦国、张磊、邹淑芳、于国槐、高茹、任媛、孙桂杰、高润岭、郭锐、李贺、陈威、张世辉、郭鑫、张领、王占龙、李根福、王喜平等。

寄语读者

亲爱的读者朋友，千里有缘一线牵，感谢您在茫茫书海中找到了本书，希望她架起你我之间学习、友谊的桥梁，希望她带您轻松步入妙趣横生的编程世界，希望她成为您成长道路上的铺路石。

软件开发技术联盟

目　录

Contents

本书光盘“自主学习系统”内容索引...XI

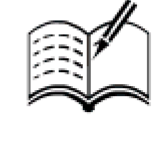
第1篇　入　门　篇

第1章　Android入门................................2
（视频讲解：22分钟）
1.1　Android概述..................................3
1.1.1　Android的定义..............................3
1.1.2　Android成功案例............................5
1.1.3　Android的版本..............................6
1.1.4　Android市场................................8
1.2　Android特性..................................8
1.2.1　开放性.....................................8
1.2.2　挣脱束缚...................................9
1.2.3　丰富的硬件.................................9
1.2.4　开发商.....................................9
1.2.5　Google应用.................................9
1.3　Android 4.3新增特性..........................9
1.3.1　用户体验..................................10
1.3.2　多用户切换与受限账户......................10
1.3.3　蓝牙......................................10
1.3.4　WiFi后台自动搜索功能......................10
1.3.5　图形......................................10
1.3.6　音频......................................11
1.3.7　流媒体加密................................11
1.3.8　通知栏....................................11
1.3.9　相机......................................11
1.3.10　拨号面板.................................11
1.3.11　键盘与输入...............................12
1.3.12　设置.....................................12
1.3.13　支持国际用户.............................12
1.3.14　新增多国语言支持.........................12
1.3.15　谷歌套件.................................12
1.3.16　其他新增特性.............................12
1.4　如何学习Android.............................13
1.4.1　如何学好Android...........................13
1.4.2　Android API文档的使用.....................13
1.5　本章小结....................................14

第2章　搭建Android开发环境......................15
（视频讲解：1小时26分钟）
2.1　搭建Android开发环境.........................16
2.1.1　Android开发准备...........................16
2.1.2　JDK的下载.................................17
2.1.3　JDK的安装与配置...........................19
2.1.4　ADT Bundle的下载..........................22
2.2　第一个Android程序...........................25
2.2.1　创建Android应用程序.......................25
2.2.2　创建AVD模拟器.............................29
2.2.3　运行Android程序...........................31
2.2.4　调试Android应用程序.......................31
2.2.5　Android应用开发流程.......................33
2.3　综合应用....................................33
2.3.1　创建一个可以运行在所有Android版本上的程序....33
2.3.2　在Android窗口中输出“你好”中文字符串........33
2.4　本章常见错误................................35
2.5　本章小结....................................35
2.6　跟我上机....................................36

第3章　认识Android模拟器........................37
（视频讲解：21分钟）
3.1　启动和删除Android模拟器.....................38
3.1.1　启动Android模拟器.........................38




Note

3.1.2 删除 Android 模拟器......39
3.2 Android 模拟器常用设置......39
3.2.1 设置语言......40
3.2.2 设置输入法......41
3.2.3 设置日期时间......42
3.3 安装和卸载程序......44
3.3.1 使用 adb 命令安装和卸载 Android 程序......44
3.3.2 通过 DDMS 管理器安装 Android 程序......46
3.3.3 在 Android 模拟器中卸载程序......47
3.4 综合应用......48
3.4.1 设置模拟器桌面背景......48
3.4.2 在 Android 模拟器中安装搜狗拼音输入法......49
3.5 本章常见错误......50
3.6 本章小结......50
3.7 跟我上机......51

第 4 章 剖析 Android 程序......52
（视频讲解：58 分钟）
4.1 Android 程序的组成......53
4.1.1 src 目录......53
4.1.2 res 目录......54
4.1.3 gen 目录及 R.java 文件......56
4.1.4 AndroidManifest.xml 文件......58
4.2 Android 程序的生命周期......59
4.3 Android 程序的基本组件......60
4.3.1 Activity（活动窗口）......60
4.3.2 BroadcastReceiver（广播接收器）......62
4.3.3 Content Provider（数据共享）......63
4.3.4 Service（服务）......65
4.4 综合应用......66
4.4.1 在 Android 程序中添加 Activity......66
4.4.2 在 Android 程序中添加 Service......67
4.5 本章常见错误......68
4.6 本章小结......68
4.7 跟我上机......68

第 5 章 Android 常用组件的使用......69
（视频讲解：2 小时 42 分钟）
5.1 Android 的 UI 界面......70
5.1.1 Android UI 界面概述......70
5.1.2 使用 XML 布局文件控制 UI 界面......70
5.1.3 在 Java 代码中控制 UI 界面......72
5.1.4 使用 XML 和 Java 代码混合控制 UI 界面......74
5.1.5 开发自定义的 View......76
5.2 文本类组件......78
5.2.1 TextView 组件......78
5.2.2 EditText 组件......81
5.2.3 AutoCompleteTextView 组件......83
5.3 按钮类组件......86
5.3.1 Button 组件......86
5.3.2 ImageButton 组件......88
5.3.3 ToggleButton 组件......89
5.4 选择类组件......91
5.4.1 RadioButton 组件......91
5.4.2 CheckBox 组件......94
5.5 列表类组件......97
5.5.1 ListView 组件......97
5.5.2 Spinner 组件......102
5.6 图像类组件......104
5.6.1 ImageView 组件......104
5.6.2 Gallery 组件......106
5.6.3 ImageSwitcher 组件......109
5.7 综合应用......111
5.7.1 实现带图标的 ListView 列表......111
5.7.2 猜猜鸡蛋放在哪只鞋子里......114
5.8 本章常见错误......117
5.9 本章小结......118
5.10 跟我上机......118

第 6 章 掌握布局管理器......119
（视频讲解：58 分钟）
6.1 线性布局管理器......120
6.2 绝对布局管理器......124

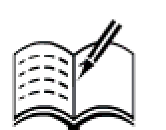


6.3 框架布局管理器 125
6.4 相对布局管理器 127
6.5 表格布局管理器 129
6.6 综合应用 132
6.6.1 我同意游戏条款 132
6.6.2 应用相对布局管理器显示软件更新提示 135
6.7 本章常见错误 136
6.8 本章小结 137
6.9 跟我上机 137

第 7 章 Android 程序调试与错误处理 139
（视频讲解：48 分钟）
7.1 输出日志信息的几种方法 140
7.1.1 Log.d 方法——输出故障日志信息 140
7.1.2 Log.e 方法——输出错误日志信息 141
7.1.3 Log.i 方法——输出程序日志信息 142
7.1.4 Log.v 方法——输出冗余日志信息 143
7.1.5 Log.w 方法——输出警告日志信息 144
7.2 Android 程序调试 146
7.3 程序异常处理 147
7.3.1 Android 程序出现异常怎么办 147
7.3.2 如何捕捉 Android 程序异常 148
7.3.3 抛出异常的两种方法 149
7.3.4 何时使用异常处理 152
7.4 综合应用 152
7.4.1 向 LogCat 视图中输出用户登录时间 152
7.4.2 使用 throw 关键字在方法中抛出异常 152
7.5 本章常见错误 153
7.6 本章小结 154
7.7 跟我上机 154

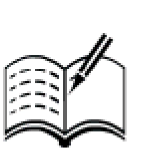
第 8 章 Activity 的使用 155
（视频讲解：1 小时 44 分钟）
8.1 Activity 入门 156
8.1.1 Activity 概述 156
8.1.2 Activity 的 4 种状态 156
8.1.3 Activity 的属性 157
8.2 Activity 的生命周期 159
8.2.1 Activity 生命周期概述 159
8.2.2 Activity 的方法 160
8.3 Activity 常用操作 164
8.3.1 创建 Activity 164
8.3.2 启动一个或多个 Activity 165
8.3.3 多个 Activity 之间的传值 169
8.3.4 关闭 Activity 172
8.4 综合应用 172
8.4.1 根据输入的生日判断星座 172
8.4.2 带选择头像的用户注册界面 176
8.4.3 仿 QQ 客户端登录界面 180
8.5 本章常见错误 183
8.6 本章小结 184
8.7 跟我上机 184

第 9 章 使用 Intent 进行通信 186
（视频讲解：56 分钟）
9.1 Intent 对象简介 187
9.1.1 Intent 对象概述 187
9.1.2 3 种不同的 Intent 传输机制 187
9.2 Intent 对象的组成 188
9.2.1 组件名称 188
9.2.2 动作 189
9.2.3 数据 190
9.2.4 种类 193
9.2.5 附加信息 194
9.2.6 标志 197
9.3 解析 Intent 对象 199
9.3.1 Intent 过滤器 199
9.3.2 通用情况 202
9.3.3 使用 Intent 匹配 203
9.4 使用 Intent 传递数据 203






9.4.1 无参数 Activity 跳转 203
9.4.2 向下一个 Activity 传递数据 203
9.5 综合应用 209
9.5.1 使用 Intent 实现直接发送短信 209
9.5.2 使用 Intent 打开网页 212
9.6 本章常见错误 214
9.7 本章小结 214
9.8 跟我上机 214


Note

第 2 篇 提 高 篇

第 10 章 Android 高级组件的使用 218
（视频讲解：1 小时 2 分钟）
10.1 日期时间类组件 219
10.1.1 AnalogClock 组件 219
10.1.2 DigitalClock 组件 220
10.2 进度条组件 221
10.2.1 ProgressBar 组件 221
10.2.2 SeekBar 组件 224
10.2.3 RatingBar 组件 227
10.3 对话框及消息提示组件 229
10.3.1 Toast 组件 229
10.3.2 Notification 组件 231
10.3.3 AlertDialog 组件 233
10.4 综合应用 238
10.4.1 显示在标题上的进度条 238
10.4.2 仿手机 QQ 登录状态显示功能 241
10.5 本章常见错误 244
10.6 本章小结 244
10.7 跟我上机 244
第 11 章 Android 中的事件处理 246
（视频讲解：20 分钟）
11.1 事件处理概述 247
11.2 处理键盘事件 247
11.3 处理触摸事件 248
11.4 手势的创建与识别 250
11.4.1 手势的创建 251
11.4.2 手势的导出 252
11.4.3 手势的识别 252
11.5 综合应用 254
11.5.1 查看手势对应分值 254
11.5.2 使用手势输入数字 255
11.6 本章常见错误 257
11.7 本章小结 257
11.8 跟我上机 257
第 12 章 数据存储技术 259
（视频讲解：44 分钟）
12.1 使用 SharedPreferences 对象存储数据 260
12.2 使用 Files 对象存储数据 268
12.2.1 openFileOutput()和 openFileInput()方法 268
12.2.2 对 Android 模拟器中的 SD 卡进行操作 271
12.3 SQLite 数据库编程 273
12.4 综合应用 277
12.4.1 遍历 Android 模拟器的 SD 卡 277
12.4.2 在 SQLite 数据库中批量添加数据 278
12.4.3 使用列表显示数据表中全部数据 281
12.5 本章常见错误 282
12.6 本章小结 283
12.7 跟我上机 283
第 13 章 Content Provider 实现数据共享 285
（视频讲解：44 分钟）
13.1 Content Provider 概述 286
13.1.1 数据模型 286





13.1.2 URI 的用法 287

13.2 Content Provider 的常用操作 287

13.2.1 查询数据 288

13.2.2 增加记录 289

13.2.3 增加新值 289

13.2.4 批量更新记录 289

13.2.5 删除记录 289

13.3 自定义 Content Provider 290

13.3.1 继承 ContentProvider 类 290

13.3.2 声明 Content Provider 292

13.4 综合应用 293

13.4.1 查询联系人 ID 和姓名 293

13.4.2 自动补全联系人姓名 294

13.5 本章常见错误 295

13.6 本章小结 296

13.7 跟我上机 296

第 14 章 图形图像处理技术 298

（视频讲解：2 小时 8 分钟）

14.1 Android 中的常用绘图类 299

14.1.1 Paint 类 299

14.1.2 Canvas 类 301

14.1.3 Bitmap 类 302

14.1.4 BitmapFactory 类 303

14.2 绘制 2D 图像 303

14.2.1 绘制几何图形 304

14.2.2 绘制文本 306

14.2.3 绘制路径 308

14.2.4 绘制图片 311

14.3 常见的图像特效 313

14.3.1 旋转图像 313

14.3.2 缩放图像 315

14.3.3 倾斜图像 317

14.3.4 平移图像 318

14.3.5 使用 BitmapShader 渲染图像 320

14.4 Android 中的动画 321

14.4.1 实现逐帧动画 321

14.4.2 实现补间动画 322

14.5 综合应用 329

14.5.1 实现带描边的圆角图片 329

14.5.2 实现放大镜效果 330

14.5.3 忐忑的精灵 332

14.6 本章常见错误 333

14.7 本章小结 334

14.8 跟我上机 334

第 15 章 利用 OpenGL 实现 3D 图形 337

（视频讲解：56 分钟）

15.1 OpenGL 概述 338

15.2 绘制 3D 图形 339

15.2.1 构建 3D 开发的基本框架 339

15.2.2 绘制一个模型 341

15.3 添加效果 345

15.3.1 应用纹理贴图 346

15.3.2 旋转 347

15.3.3 光照效果 349

15.3.4 透明效果 351

15.4 综合应用 352

15.4.1 绘制一个不断旋转的金字塔 352

15.4.2 使用 Android 机器人对立方体进行纹理贴图 354

15.5 本章常见错误 356

15.6 本章小结 356

15.7 跟我上机 356

第 16 章 多媒体应用开发 359

（视频讲解：50 分钟）

16.1 音频的播放 360

16.1.1 使用 MediaPlayer 播放音频 360

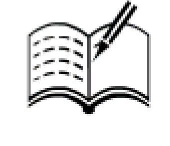
16.1.2 使用 SoundPool 播放音频 365

16.2 视频的播放 368

16.2.1 使用 VideoView 组件播放视频 368

16.2.2 使用 MediaPlayer 和 SurfaceView 播放视频 370




Note

16.3 综合应用....374

16.3.1 为游戏界面添加背景音乐和按键音....374

16.3.2 制作开场动画....379

16.4 本章常见错误....380

16.5 本章小结....381

16.6 跟我上机....381

第 17 章 线程与消息处理....383

（视频讲解：35 分钟）

17.1 多线程的基本操作....384

17.1.1 创建线程....384

17.1.2 开启线程....385

17.1.3 线程的休眠....385

17.1.4 中断线程....385

17.2 Handler 消息传递机制....388

17.2.1 循环者——Looper....388

17.2.2 消息处理类——Handler....390

17.2.3 消息类——Message....391

17.3 综合应用....392

17.3.1 开启新线程实现电子广告牌....392

17.3.2 多彩的霓虹灯....394

17.3.3 简易打地鼠游戏....396

17.4 本章常见错误....398

17.5 本章小结....399

17.6 跟我上机....399

第 18 章 网络编程技术....401

（视频讲解：1 小时 6 分钟）

18.1 通过 HTTP 访问网络....402

18.1.1 使用 HttpURLConnection 访问网络....402

18.1.2 使用 HttpClient 访问网络....410

18.2 使用 WebView 显示网页....416

18.2.1 使用 WebView 组件浏览网页....416

18.2.2 使用 WebView 组件加载 HTML 代码....418

18.2.3 让 WebView 组件支持 JavaScript....419

18.3 综合应用....421

18.3.1 打造功能实用的网页浏览器....421

18.3.2 获取天气预报....424

18.4 本章常见错误....426

18.5 本章小结....426

18.6 跟我上机....427

第 19 章 Service 服务的使用....429

（视频讲解：40 分钟）

19.1 Service 概述....430

19.1.1 Service 的分类....430

19.1.2 Service 类的重要方法....430

19.1.3 Service 的声明....432

19.2 Started Service 的使用....433

19.2.1 继承 IntentService 类....434

19.2.2 继承 Service 类....435

19.2.3 启动服务....436

19.2.4 停止服务....437

19.3 Bound Service 的使用....437

19.3.1 继承 Binder 类....438

19.3.2 使用 Messenger 类....440

19.3.3 绑定到服务....442

19.4 管理 Service 的生命周期....443

19.5 综合应用....443

19.5.1 继承 IntentService 输出当前时间....443

19.5.2 继承 Service 输出当前时间....446

19.5.3 继承 Binder 类绑定服务显示时间....447

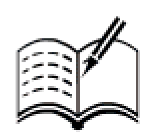
19.5.4 使用 Messenger 类绑定服务显示时间....450

19.6 本章常见错误....453

19.7 本章小结....453

19.8 跟我上机....453





第3篇 实 战 篇

第20章 Android游戏——数独游戏 456
（视频讲解：28分钟）
20.1 需求分析 457
20.2 程序开发及运行环境 457
20.3 程序文件夹组织结构 457
20.4 公共资源文件 458
20.4.1 字符串资源文件 458
20.4.2 数组资源文件 458
20.4.3 颜色资源文件 459
20.5 游戏主窗体设计 459
20.5.1 设计系统主窗体布局文件 460
20.5.2 为界面中的按钮添加监听事件 462
20.5.3 绘制数独游戏界面 463
20.5.4 数独游戏的实现算法 468
20.6 虚拟键盘模块设计 472
20.6.1 设计虚拟键盘布局文件 473
20.6.2 在虚拟键盘中显示可以输入的数字 473
20.7 游戏设置模块设计 475
20.7.1 设计游戏设置布局文件 476
20.7.2 设置是否播放背景音乐和显示提示 476
20.7.3 控制背景音乐的播放与停止 477
20.8 关于模块设计 477
20.8.1 设计关于窗体布局文件 478
20.8.2 显示关于信息 478
20.9 将程序安装到Android手机上 479
20.10 本章小结 480
第21章 Android应用——家庭理财通 481
（视频讲解：46分钟）
21.1 需求分析 482
21.2 系统设计 482
21.2.1 系统目标 482
21.2.2 系统功能结构 482
21.2.3 系统业务流程图 482
21.2.4 系统编码规范 483
21.3 系统开发及运行环境 484
21.4 数据库与数据表设计 485
21.4.1 数据库分析 485
21.4.2 创建数据库 485
21.4.3 创建数据表 486
21.5 系统文件夹组织结构 487
21.6 公共类设计 488
21.6.1 数据模型公共类 488
21.6.2 Dao公共类 490
21.7 登录模块设计 495
21.7.1 设计登录布局文件 495
21.7.2 登录功能的实现 496
21.7.3 退出登录窗口 497
21.8 系统主窗体设计 497
21.8.1 设计系统主窗体布局文件 498
21.8.2 显示各功能窗口 499
21.8.3 定义文本及图片组件 500
21.8.4 定义功能图标及说明文字 501
21.8.5 设置功能图标及说明文字 501
21.9 收入管理模块设计 503
21.9.1 设计新增收入布局文件 503
21.9.2 设置收入时间 507
21.9.3 添加收入信息 508
21.9.4 重置新增收入窗体中的各个控件 509
21.9.5 设计收入信息浏览布局文件 509
21.9.6 显示所有的收入信息 510
21.9.7 单击指定项时打开详细信息 511



Note

21.9.8 设计修改/删除收入布局文件..512
21.9.9 显示指定编号的收入信息........515
21.9.10 修改收入信息........................517
21.9.11 删除收入信息........................518
21.10 便签管理模块设计..................518
21.10.1 设计新增便签布局文件.........519
21.10.2 添加便签信息.........................520
21.10.3 清空便签文本框.....................521
21.10.4 设计便签信息浏览布局文件..521
21.10.5 显示所有的便签信息.............523
21.10.6 单击指定项时打开详细信息...525
21.10.7 设计修改/删除便签布局文件...526
21.10.8 显示指定编号的便签信息.......528
21.10.9 修改便签信息.........................528
21.10.10 删除便签信息.......................529
21.11 系统设置模块设计..................529
21.11.1 设计系统设置布局文件...........530
21.11.2 设置登录密码.........................531
21.11.3 重置密码文本框.....................531
21.12 开发常见问题与解决...............532
21.12.1 程序在装有 Android 系统的手机上无法运行......................532
21.12.2 无法将最新修改在 Android 模拟器中体现.........................532
21.12.3 退出系统后还能使用记录的密码登录................................532
21.13 本章小结.................................533

本书光盘“自主学习系统”（各类学习资源库）

内容索引

说明：

亲爱的读者朋友，熟练掌握一门编程工具，一本书是远远不够的。为了方便您深入学习、拓展视野，我们开发整理了海量的学习资源库，即配书光盘中的“自主学习系统”，内容有6大部分：

1. **实例资源库：**包括**278个Android编程实例，1062个Java编程实例，各种类型一应俱全**，无论学习这本书的哪一章节，都可以从中找到相关的多种实例加以实践，相信对深入学习极有帮助。

2. **模块资源库：**包括了最常用的**17个Java经典模块分析**，它们既可作为**综合应用实例学习，又可移植到相关应用中**，进而避免重复劳动，提高工作效率。

3. **项目（案例）资源库：**包括**17个Java项目开发案例，从需求分析、系统设计、模块分析到代码实现**，几乎全程展现了项目开发的整个过程。

4. **任务（训练）资源库：**共计**99个Android编程实践任务**，读者可以自行实践练习，还可以到对应的网站上寻找答案。

5. **能力测试资源库：**列举了**616道Java能力测试题目，包括编程基础能力测试、数学及逻辑思维能力测试、面试能力测试、编程英语能力测试**，便于读者自我测试。

6. **编程人生：**精选了**23个IT励志故事**，希望读者朋友从这些IT成功人士的经历中**汲取精神力量**，让这些经历成为您不断进取、勇攀高峰的强大精神动力。

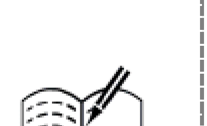
第1部分　实例资源库
（1340个完整实例分析）

Android部分

Android模拟器应用
- 创建一个Android模拟器
- 启动Android模拟器
- 删除Android模拟器
- 为Android模拟器设置语言
- 为Android模拟器设置输入法
- 为Android模拟器设置日期时间
- 使用adb命令安装Android程序
- 使用adb命令卸载Android程序
- 通过DDMS管理器安装Android程序
- 在Android模拟器中卸载程序
- 在Android模拟器中安装搜狗拼音输入法
- 设置模拟器桌面背景
- 设置使用24小时格式的时间
- 使用Android模拟器发送短信
- 使用Android模拟器拨打电话
- 查看Android模拟器中正在运行的服务

界面布局及菜单设计
- 使用XML布局文件实现游戏的开始界面
- 通过Java代码实现游戏的进入界面
- 使用XML和Java代码混合控制UI界面
- 通过自定义View组件实现Activity界面的切换
- 使用线性布局管理器布局Android界面
- 使用绝对布局固定组件的位置
- 使用框架布局居中显示层叠的正方形
- 使用相对布局管理器布局多个组件相对位置



Note

- 使用表格布局管理器布局用户的登录界面
- 我同意游戏条款界面布局
- 仿微信全民打飞机游戏的用户许可协议界面
- 应用相对布局显示软件更新提示
- 使用表格布局与线性布局实现分类工具栏
- 布局个性游戏开始界面
- 通过自定义 View 组件实现跟随手指的小兔子
- 在窗体上绘制一只地鼠
- 布局用户搜索界面
- 用于改变文字颜色的上下文菜单
- 实现带子菜单的选项菜单
- 创建一组只能单选的选项菜单
- 对选项菜单进行国际化
- 隐藏动作栏
- 自定义动作项
- 为按钮提供隐藏和显示动作栏标题
- 在动作栏中添加和删除选项卡
- 在动作栏中增加“查找”动作视图
- 在动作栏中添加“设置”图标
- 重新设置 Icon 图标

Android 常用组件应用

- 应用 TextView 显示多种样式的文本
- 使用 EditText 组件实现用户注册信息的输入
- 为文本框组件添加滚动条
- 使用文本框控件记录历史查询记录
- 添加两个按钮并为其设置单击事件监听器
- 使用 ImageButton 组件实现图片按钮
- 获取 ToggleButton 按钮上的当前文本
- 使用 AutoCompleteTextView 组件实现自动提示功能
- 添加选择性别的单选按钮
- 选择爱好的复选按钮组
- 通过数组资源为 ListView 设置列表项
- 使用适配器为 ListView 设置列表项
- 应用 ListView 显示带头、脚视图的列表
- 通过继承 ListActivity 实现列表
- 显示列表选择框并获取其选择项
- 使用 ImageView 显示图像
- 使用 Gallery 组件显示图片列表
- 使用 ImageSwitcher 组件实现简单图片查看器
- 改进后的图片查看器
- 通过 GridView 显示照片列表
- 仿 Windows 7 图片预览窗格效果
- 幻灯片式图片浏览器
- 实现带图标的 ListView 列表
- 实现图标在上，文字在下的 ListView

使用文本框控件记录历史查询记录

- 在屏幕中显示模拟时钟
- 应用日期、时间拾取器选择日期和时间
- 使用 DigitalClock 组件显示详细时间
- 显示计时器
- 定时关闭当前窗口
- 在屏幕中显示水平进度条和圆形进度条
- 在屏幕中显示拖动条 doc
- 在屏幕中显示星级评分条
- 显示消息提示框
- 在状态栏上显示通知
- 发送一个自定义声音提示的通知
- 多种形式的列表对话框
- 弹出询问是否退出的对话框
- 选择颜色的单选列表对话框
- 应用 AlertDialog 实现自定义的登录对话框
- 询问是否评价的自定义对话框
- 显示在标题上的进度条
- 仿手机 QQ 登录状态显示功能
- 设置定时启动的闹钟
- 设置一个 BroadcastReceiver 闹钟
- 应用 AlarmManager 实现定时更换壁纸功能
- 在屏幕中添加选项卡

Android 程序调试

- 使用 Log.d 方法输出 Debug 日志信息
- 使用 Log.e 方法输出错误日志信息
- 使用 Log.i 方法输出程序日志信息
- 使用 Log.v 方法输出冗余日志信息
- 使用 Log.w 方法输出警告日志信息
- 使用 try…catch 语句捕获 Android 程序异常
- 使用 throws 关键字抛出异常
- 使用 throw 关键字抛出异常
- 向 LogCat 视图中输出用户登录时间
- 使用 throw 关键字在方法中抛出异常

Activity 窗口设计

- 在 Android 程序中添加 Activity
- 在 Android 程序中添加 Service
- 启动和关闭 Activity
- 在多个 Activity 之间实现相互传值
- 用户注册中的返回上一步功能
- 根据输入的性别和身高计算标准体重
- 根据分数显示优、良、中、差的评价
- 根据输入的生日判断星座
- 带选择头像的用户注册界面



- 实现带选择所在城市的用户注册界面
- 实现带选择商品类别的商品信息添加
- 仿 QQ 客户端登录界面
- 实现一个泡泡龙游戏的关于功能
- 显示标题列表及选定标题对应的详细内容
- 实现古诗欣赏程序
- 带查看原图的图像浏览器
- 应用 Fragment 实现图片查看器
- 应用 Fragment 实现新闻浏览

Intent 通信应用

- 通过 Intent 实现拨打电话功能
- 通过 Intent 实现发送短信功能
- 将字符串数据传递到打开的 Activity 中
- 得到新打开 Activity 关闭后返回的数据
- 使用 Intent 查看通讯录信息
- 使用 Intent 修改通讯录信息
- 使用 Intent 实现直接发送短信
- 使用 Intent 打开网页
- 使用 Intent 实现返回系统 Home 桌面
- 当接收到短信时给出提示信息
- 接收短信后显示短信号码
- 接收短信后显示短信内容
- 用户单击按钮时显示电池剩余电量
- 当电池电量低于 10%时给出提
- 安装新应用后给出提示的功能

数据存储技术

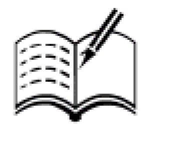
- 使用 SharedPreferences 保存用户输入的用户名和密码
- 使用 SharedPreferences 保存用户输入值
- 获取 SharedPreferences 中保存的值
- 使用 SharedPreferences 在 Activity 间传递整数值
- 使用 SharedPreferences 在 Activity 间传递布尔值
- 使用内部存储保存用户输入的用户名和密码
- 显示内部存储文件位置的绝对路径
- 在 SD 卡上创建文件
- 使用 SQLite 数据库保存用户输入的用户名和密码
- 在 SQLite 数据库中批量添加数据
- 使用列表显示数据表中全部数据
- 使用列表逆序显示数据表中全部数据
- 判断获得的 SD 卡内容是否是文件夹
- 显示文件和文件夹的创建时间
- 遍历 Android 模拟器的 SD 卡
- 复制图片到 SD 卡上
- 使用 Content Provider 查询数据
- 使用 Content Provider 添加记录
- 显示联系人 ID 和公司信息
- 使用 Content Provider 删除记录
- 系统内置联系人的使用
- 查询联系人的 ID 和姓名
- 自动补全联系人姓名
- 显示联系人姓名和电话
- 根据电话号码查找联系人

图形图像处理技术

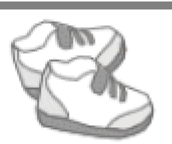
- 绘制以渐变色填充的矩形
- 创建绘图画布并绘制带阴影的矩形
- 绘制渐变色填充的圆形
- 绘制 5 个不同颜色的圆形
- 绘制一个游戏对白界面
- 绘制路径及绕路径文字
- 绘制 Android 的机器人
- 在屏幕上绘制小房子
- 在屏幕上绘制彩色字符串
- 绘制一个随机数字组成的验证码
- 使用 Matrix 旋转图像
- 使用 Matrix 缩放图像
- 使用 Matrix 倾斜图像
- 使用 Matrix 平移图像
- 显示平铺背景和椭圆形的图片
- 旋转、平移、缩放和透明度渐变的补间动画
- 绘制带描边的圆角矩形图片
- 带描边的圆形图片
- 实现放大镜效果
- 实现探照灯效果
- 实现闪烁的星星
- 实现在夜空中同时有多颗星星闪烁的效果
- 来回捕食的小鱼
- 飞舞的蝴蝶
- 简易涂鸦板
- 在 GridView 中显示 SD 卡上的全部图片

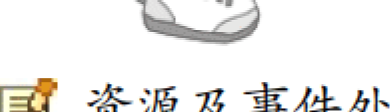
3D 及多媒体开发

- 绘制一个 6 个面采用不同颜色的立方体
- 为立方体进行纹理贴图
- 不断旋转的立方体
- 为立方体添加光照效果
- 透明且旋转的立方体
- 绘制一个不断旋转的金字塔
- 使用 Android 机器人对立方体进行纹理贴图
- 绘制一个三棱锥
- 包括播放、暂停继续和停止功能的音乐播放器
- 带音量控制的音乐播放器
- 使用 SoundPool 播放音频
- 使用 VideoView 组件播放视频
- 使用 MediaPlayer 和 SurfaceView 播放视频
- 为游戏界面添加背景音乐和按键音
- 为 E、S、D 和 F 键添加按键音
- 制作开场动画
- 控制相机拍照
- 在拍摄照片上添加拍照日期
- 在拍摄照片上添加边框









资源及事件处理
- 使用字符串资源设置界面中的文字
- 通过字符串资源显示游戏对白
- 使用颜色资源设置文字颜色
- 使用颜色资源设置窗体的背景颜色
- 逐渐加宽的彩虹桥背景
- 通过尺寸资源将文字逐个放大
- 使用 9-Patch 图片实现不失真按钮背景
- 使用 9-Patch 图片实现登录和退出按钮
- 自定义复选按钮的样式
- 背景半透明效果的游戏开始界面
- 应用样式资源改变文字的样式
- 应用主题资源给所有窗口添加背景
- 从 XML 文件中读取客户信息
- 显示用户单击的按键
- 判断是否为系统按键
- 屏蔽物理键盘中的后退键
- 显示短时间和长时间单击按钮信息
- 当用户触摸屏幕时显示提示信息
- 显示用户触摸时持续的时间
- 显示用户触摸屏幕位置
- 识别用户输入的手势
- 查看手势对应分值
- 使用手势输入数字
- 根据输入手势拨打电话
- 单击增加音量键时显示提示信息

多线程编程
- 通过实现 Runnable 接口创建、开启、休眠和中断线程
- 在日志窗口中每隔 1 秒显示一个文字
- 每隔 1 分钟更换一次桌面背景
- 创建 Handler 对象发送并处理消息
- 使用线程和消息传递机制实现水平移动的图标
- 开启新线程实现电子广告牌
- 多彩的霓虹灯
- 海滩捉蟹游戏
- 在屏幕上来回移动的气球
- 开启新线程播放背景音乐

网络开发应用
- 向服务器发送 GET 请求
- 向服务器发送 POST 请求
- 使用 HttpClient 向服务器发送 GET 请求
- 使用 HttpClient 向服务器发送 POST 请求
- 通过 GET 请求发送中文参数
- 使用 WebView 浏览网页
- 使用 WebView 加载 HTML 代码
- 让 WebView 允许执行 JavaScript
- 从指定网站下载文件
- 使用 ImageView 显示从网络上获取的图片
- 打造功能实用的网页浏览器
- 获取天气预报
- 继承 IntentService 输出当前时间
- 继承 Service 输出当前时间
- 继承 Binder 类绑定服务显示时间
- 使用 Message 类绑定服务显示时间
- 视力保护程序
- 获得当前模拟器支持的全部位置源名称
- 获得 GPS 位置源的精度和耗电量
- 获得经纬度信息
- 获取当前位置的海拔信息
- 获得谷歌地图 API 密钥
- 在地图上标记天府广场的位置

Android 游戏开发
- 猜猜鸡蛋放在哪只鞋子里
- 忐忑的精灵
- 迷途奔跑的野猪
- 简易打地鼠游戏
- 数独游戏——主窗体设计
- 数独游戏——虚拟键盘模块设计
- 数独游戏——游戏设置模块设计
- 数独游戏——关于模块设计

Android 综合应用
- 家庭理财通——登录模块设计
- 家庭理财通——系统主窗体设计
- 家庭理财通——新增收入设计
- 家庭理财通——收入信息浏览设计
- 家庭理财通——修改删除收入设计
- 家庭理财通——新增便签设计
- 家庭理财通——便签信息浏览设计
- 家庭理财通——修改删除便签设计
- 家庭理财通——系统设置模块设计

Java 部分

开发环境的应用
- 下载 JDK 开发工具包
- 把 JDK 工具包安装到指定磁盘
- 设置 JDK 的环境变量
- 验证 Java 开发环境
- 下载并安装 JRE 执行环境
- 编程输出星号组成的等腰三角形
- 下载最新的 Eclipse
- 为最新的 Eclipse 安装中文语言包
- 活用 Eclipse 的工作空间



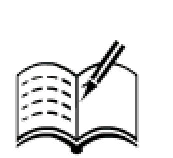
- 在 Eclipse 项目中编程输出字符表情
- 为 Eclipse 添加新的 JDK 环境
- 设置 Eclipse 中文 API 提示信息
- 为项目添加类库
- 使当前项目依赖另一个项目
- 安装界面设计器
- 设计 Windows 系统的运行对话框界面
- 设计计算器程序界面
- 设计进销存管理系统的关于界面
- JDK 1.5 的安装与配置
- Tomcat 5.5 的安装
- 配置 Windows 2000+SQL Server 2000+Tomcat 运行
- 配置 Windows 2000+Oracle +Tomcat 运行环境
- 配置 Windows 2000+Access +Tomcat 运行环境
- 配置 Windows 2000+MySQL+ Tomcat 运行环境
- 配置 Windows XP 2003+SQL Server 2000+Tomcat 运行环境
- 在 Linux 下安装 JDK 1.5
- 在 Linux 下配置 Tomcat 服务器
- 配置 Linux+MySQL+ Tomcat 运行环境
- 配置 Windows+Resin 运行环境
- 配置 Linux+Resin 运行
- 安装与配置 WebLogic 服务器
- WebLogic 中 SQL Server 2000 的 JDBC 连接池配置
- 应用 Dreamweaver 开发 JSP 程序
- 配置 Windows 2000+MySQL+ Tomcat 运行环境
- 应用 MyEclipse 开发 JSP 程序
- 应用 NetBeans 开发 JSP 程序
- Java 基础应用
- 输出错误信息与调试信息
- 从控制台接收输入字符
- 重定向输出流实现程序日志
- 自动类型转换与强制类型转换
- 加密可以这样简单（位运算）
- 用三元运算符判断奇数和偶数
- 更精确的使用浮点数
- 不用乘法运算符实现 2×16
- 实现两个变量的互换（不借助第 3 个变量）
- 判断某一年是否为闰年
- 验证登录信息的合法性
- 为新员工分配部门
- 用 Switch 语句根据消费金额计算折扣
- 判断用户输入月份的季节
- 使用 while 与自增运算符循环遍历数组
- 使用 for 循环输出杨辉三角
- 使用嵌套循环在控制台上输出九九乘法表
- 用 while 循环计算 1+1/2!+1/3!…1/20!
- for 循环输出空心的菱形
- foreach 循环优于 for 循环
- 终止循环体
- 循环体的过滤器
- 循环的极限
- 数组与集合的应用
- 获取一维数组最小值
- 将二维数组中的行列互换
- 利用数组随机抽取幸运观众
- 用数组设置 JTable 表格的列名与列宽
- 数组的下标界限
- 按钮控件数组实现计数器界面
- 复选框控件数组
- 用数组把字符串反转
- 使用选择排序法
- 使用冒泡排序法
- 使用快速排序法
- 使用直接插入法
- 使用 Sort 方法对数组进行排序
- 反转数组中元素的顺序
- 用动态数组保存学生姓名
- 用 List 集合传递学生信息
- 用 TreeSet 生成不重复自动排序随机数组
- Map 映射集合实现省市级联选择框

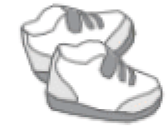
字符串处理技术

- 把数字格式化为货币字符串
- 格式化当前日期
- 货币金额大写格式
- String 类格式化当前日期
- 字符串大小写转换
- 字符与 Unicode 码的转换
- 判断用户名是否正确
- 用户名排序
- 判断网页请求与 FTP 请求
- 判断文件类型
- 判断字符串是否为数字
- 验证 IP 地址的有效性
- 鉴别非法电话号码
- 根据标点符号对字符串进行分行
- 将字符串的每个字符进行倒序输出
- 获取字符串中汉字的个数
- 批量替换某一类字符串
- 把异常与错误信息显示到窗体中
- 从字符串中分离文件路径、文件名及扩展名
- 判断手机号的合法性
- 用字符串构建器追加字符
- 去掉字符串中的所有空格
- 汉字与区位码的转换

Java 中类的定义

- 自定义图书类
- 温度单位转换工具
- 域的默认初始化值
- 编写同名的方法
- 构造方法的应用
- 单例模式的应用
- 祖先的止痒药方
- 统计图书的销售量
- 汉诺塔问题求解
- 不能重写的方法
- 将字符串转换成整数




Note

- 整数进制转换器
- 查看数字的取值范围
- ASCII 编码查看器
- Double 类型的比较
- 经理与员工的差异
- 重写父类中的方法
- 计算几何图形的面积
- 提高产品质量的方法
- 简单的汽车销售商场
- 两只完全相同的宠物
- 简化 equals()方法的重写
- 重新计算对象的哈希码
- 简化 hashCode()方法的重写
- 使用字符串输出对象
- 简化 toString()方法的重写
- Java 对象的假克隆
- Java 对象的浅克隆
- Java 对象的深克隆
- 序列化与对象克隆
- 深克隆效率的比较
- transient 关键字的应用
- 使用 sort()方法排序
- 简化 compareTo()方法的重写
- 策略模式的简单应用
- 适配器模式的简单应用
- 普通内部类的简单应用
- 局部内部类的简单应用
- 匿名内部类的简单应用
- 静态内部类的简单应用

多线程技术

- 新建无返回值的线程
- 查看线程的运行状态
- 查看 JVM 中的线程名
- 查看和修改线程名称
- 查看和修改线程优先级
- 使用守护线程
- 休眠当前线程
- 终止指定线程
- 线程的插队运行
- 非同步的数据读写
- 使用方法实现线程同步
- 使用代码块实现线程同步
- 使用特殊域变量实现线程同步
- 使用重入锁实现线程同步
- 使用线程局部变量实现线程同步
- 简单的线程通信
- 简单的线程死锁
- 解决线程的死锁问题
- 使用阻塞队列实现线程同步
- 新建有返回值的线程
- 使用线程池优化多线程编程
- Object 类中线程相关方法
- 哲学家就餐问题
- 使用信号量实现线程同步
- 使用原子变量实现线程同步
- 使用事件分配线程更新 Swing 控件
- 使用 SwingWork 类完成耗时操作

反射与异常处理

- 实例化 Class 类的 5 种方式
- 获得 Class 对象表示实体的名称
- 查看类的声明
- 查看类的成员
- 按继承层次对类排序
- 查看内部类信息
- 动态设置类的私有域
- 动态调用类中方法
- 动态实例化类
- 创建长度可变的数组
- 利用反射重写 toString()方法
- 反射与动态代理
- 算数异常
- 数组存值异常
- 数组下标越界异常
- 空指针异常
- 类未发现异常
- 非法访问异常
- 文件未发现异常
- 数据库操作异常
- 方法中抛出异常
- 方法上抛出异常
- 自定义异常类
- 捕获单个异常
- 捕获多个异常

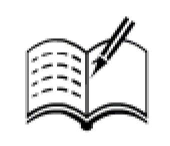
枚举与泛型的应用

- 查看枚举类型的定义
- 枚举类型的基本特性
- 增加枚举元素的信息
- 选择合适的枚举元素
- 高效的枚举元素集合
- 高效的枚举元素映射
- 遍历枚举接口的元素
- 简单的文件合并工具
- 自定义非泛型栈结构
- 使用泛型实现栈结构
- 自定义泛型化数组类
- 泛型方法与数据查询
- 泛型化方法与最小值
- 泛型化接口与最大值
- 使用通配符增强泛型
- 泛型化的折半查找法

编程常用类

- 简单的数字时钟
- 简单的电子时钟
- 简单的模拟时钟
- 简单的公历万年历
- 查看生日相关信息
- 日期格式有效性判断
- 常见日期格式使用
- 查看本地时区
- 简单的时区转换工具
- 查看常用系统属性
- 重定向标准输出
- 计算程序运行时间
- 从控制台输入密码
- 角度和弧度的转换
- 三角函数的使用
- 反三角函数的使用
- 双曲函数的使用
- 指数与对数运算
- 高精度整数运算
- 高精度浮点运算
- 七星彩号码生成器
- 大乐透号码生成器
- 监视 JVM 内存状态
- 启动默认文本工具
- 简单的截图软件





Commons 组件

- 数组元素的增加
- 数组元素的删除
- 生成随机字符串
- 序列化与反序列化
- 分数的常见运算
- 整数取值范围判断
- 描述统计学应用
- 绘制简单直方图
- 一元线性回归计算
- 实数矩阵的运算
- 复数的常见运算
- T 分布常用计算
- 简化文件（夹）删除
- 简化文件（夹）复制
- 简化文件（夹）排序
- 简化文件（夹）过滤
- 简化文件的读写操作
- 设置 JavaBean 简单属性
- 设置 JavaBean 级联属性
- 动态生成 JavaBean
- 复制 JavaBean 属性
- 动态排序 JavaBean
- 优雅的 JDBC 代码
- 结果集与 Bean 列表
- 编写 MD5 查看器
- 基于 Base64 编码
- 基于 Base64 解码
- 发送简单 Email
- 发送带附件 Email
- 读取 XML 文件属性

表单及表单元素的应用

- 获取文本框 编辑框 隐藏域的值
- 获取下拉列表 菜单的值
- 获取复选框的值
- 获取单选按钮的值
- 把数据库中的记录显示到下拉列表中
- 将数组中的数据添加到下拉列表中
- 级联菜单
- 修改数据时下拉列表的默认值为数据库中原数据信息
- 可以输入文字的下拉列表
- 根据下拉列表的值显示不同控件
- 根据数据表结构自动生成数据录入页面
- 投票信息一次性设置
- 自动计算金额
- 设置文本框的只读属性
- 让您的密码域更安全
- 限制多行文本域输入的字符个数
- 不提交表单获取单选按钮的值
- 选中单选按钮后显示其他表单元素
- 防止表单在网站外部提交
- 同一个页中的多表单提交
- 标签及设计模式专题

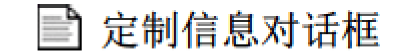
利用〈c：forEach〉循环标签实现数据显示

- 利用 EL 表达式语言实现页面逻辑处理简单化
- 自定义文件下载标签
- 自定义图片浏览标签
- 自定义数据查询标签
- 应用 HQL 检索方式查询数据
- 应用 QBC 检索方式查询数据
- 应用一对一关联实现级联添加数据
- 应用一对多关联实现级联操作
- 使用本地 SQL 检索
- DispathAction 类实现用户查询
- LookupDispatchAction 类实现用户管理
- 利用 Token 令牌机制处理用户重复提交
- 利用 Validator 验证框架处理用户登录
- 解决用户提交的中文乱码
- 利用动态 FormBean 实现对用户的操作
- Struts 与 Hibernate 结合实现数据添加和查询
- 在 Spring 中的表单控制器中实现验证处理
- 利用表单控制器实现数据添加操作
- 利用 Spring 中的多方法控制器实现数据查询和删除操作
- 通过 Spring+Hibernate 框架实现大批量数据添加

窗体设计与特效

- 控制窗体加载时的位置
- 设置窗体在屏幕中的位置
- 从上次关闭位置启动窗体
- 始终在桌面最顶层显示的窗体
- 设置窗体大小
- 根据桌面大小调整窗体大小
- 自定义最大化、最小化和关闭按钮
- 禁止改变窗体的大小
- 指定窗体标题栏图标
- 拖动没有标题栏的窗体
- 取消窗体标题栏与边框
- 设置闪烁的标题栏
- 设置窗体背景颜色为淡蓝色
- 实现带背景图片的窗体
- 使背景图片自动适应窗体的大小
- 背景为渐变色的主界面
- 随机更换窗体背景
- 椭圆形窗体界面
- 钻石形窗体
- 创建透明窗体
- 模态对话框与非模态对话框
- 信息提示对话框
- 设置信息提示对话框的图标
- 文件选择对话框指定数据库备份文件
- 指定打开话框的文件类型
- 文件的保存对话框
- 为保存对话框设置默认文件名
- 支持图片预览的文件选择对话框
- 颜色选择对话框
- 信息输入对话框





Note

- 定制信息对话框
- 创建内部子窗体
- 使子窗体最大化显示
- 对子窗体进行平铺排列
- 禁用 MDI 窗体控制栏中“最大化”按钮
- 右下角弹出信息窗体
- 淡入淡出的窗体
- 窗体顶层的进度条
- 设置窗体的鼠标光标
- 窗体抖动
- 窗体标题显示计时器
- 动态展开窗体
- 仿 QQ 隐藏窗体
- 窗体百叶窗登场特效
- 关闭窗体打开网址
- Nimbus 外观
- 本地系统外观
- 分割的窗体界面
- 圆周运动的窗体

窗口与导航条设计

- 打开新窗口显示广告信息
- 自动关闭的广告窗口
- 弹出窗口居中显示
- 打开新窗口显示详细信息
- 弹出窗口的 Cookie 控制
- 为弹出的窗口加入关闭按钮
- 关闭弹出窗口时刷新父窗口
- 关闭 IE 主窗口时，不弹出询问对话框
- 弹出网页模式对话框
- 弹出全屏显示的网页（模式）对话框
- 网页拾色器
- 日期选择器
- 全屏显示无边框有滚动条窗口
- 应用 JavaScript 实现指定尺寸的无边框窗口
- 应用 CSS+DIV 实现无边框窗口
- 带图标的文字导航条
- Flash 导航条
- 图片按钮导航条
- 导航条的动画效果
- 不用图片实现质感导航条
- 二级导航菜单
- 半透明背景的下拉菜单
- 弹出式下拉菜单
- 展开式导航条
- 收缩式导航菜单
- 树状导航菜单

应用与控制

- 将表单数据输出到 Word
- 将查询结果输出到 Word
- 通过 ODBC 访问 Excel
- 利用 Java Excel 访问 Excel
- 在 JSP 页面通过按钮打开新的 Excel 文件
- 将查询结果导出到 Excel
- 导出到 Access 数据库中
- 导出到 Excel 数据库中
- 在 JSP 中压缩 ZIP 文件
- 在 JSP 中解压缩 ZIP 文件
- 利用 Spring 生成 Excel 工作表
- 利用 Spring 生成 PDF 文件

基本控件应用

- 框架容器的背景图片
- 更多选项的框架容器
- 拦截事件的玻璃窗格
- 简单的每日提示信息
- 震动效果的提示信息
- 边框布局的简单应用
- 流式布局的简单应用
- 网格布局的简单应用
- 制作圆形布局管理器
- 制作阶梯布局管理器
- 可以打开网页的标签
- 密码域控件简单应用
- 文本域设置背景图片
- 文本区设置背景图片
- 简单的字符统计工具
- 能预览图片的复选框
- 简单的投票计数软件
- 单选按钮的简单应用
- 能显示图片的组合框
- 使用滑块来选择日期
- 模仿记事本的菜单栏
- 自定义纵向的菜单栏
- 复选框与单选框菜单
- 包含图片的弹出菜单
- 工具栏的实现与应用
- 自定义软件安装向导
- 查看系统支持的外观
- 制作软件的闪屏界面
- 自定义系统托盘图标
- 使用撤销与重做功能

复合数据类型控件应用

- 修改列表项显示方式
- 修改列表项选择模式
- 列表项的全选与不选
- 列表元素与提示信息
- 监听列表项单击事件
- 监听列表项双击事件
- 实现自动排序的列表
- 列表项的增加与删除
- 查找特定的列表元素
- 包含边框的列表元素
- 包含图片的列表元素
- 可以预览字体的列表
- 表头与列的高度设置
- 调整表格各列的宽度
- 设置表格的选择模式
- 为表头增添提示信息
- 单元格的粗粒度排序
- 实现表格的查找功能
- 在表格中应用复选框
- 删除表格中选中的行
- 实现表格的分页技术
- 为单元格绘制背景色
- 实现表格的栅栏效果
- 单元格的细粒度排序
- 编写中国省市信息树
- 树控件常用遍历方式
- 自定义树节点的图标
- 监听节点的选择事件
- 设置树控件选择模式
- 查看节点的各种状态
- 在树控件中增加节点
- 在树控件中删除节点
- 在树控件中查找节点



- 自定义树节点的外观
- 树节点增加提示信息
- 双击编辑树节点功能

其他高级控件应用

- 自定义文档标题的样式
- 文档中显示自定义图片
- 检查代码中括号是否匹配
- 描红显示 100 内的质数
- 自定义 RTF 文件查看器
- 编写简单的浏览器
- 支持超链接的浏览器
- 高亮用户指定的关键字
- 只能输入整数的文本域
- 强制输入合法的整数
- 使用微调控件调整时间
- 使用微调控件浏览图片
- 显示完成情况的进度条
- 监听进度条的变化事件
- 进度监视器控件的应用
- 监视文件读入的进度
- 分割面板的简单应用
- 为选项卡增加快捷键
- 为选项卡标题设置图标
- 记录选项卡的访问状态

控件特效与自定义控件

- 实现标签控件的立体边框
- 实现按钮控件边框留白
- 实现文本域控件的浮雕化边框
- 为文本框控件添加 LineBorder 线形边框
- 控件的纯色边框与图标边框
- 实现带标题边框的面板容器
- 指定字体的标题边框
- 嵌套的标题边框
- 带图标边框的标题边框
- 文本框的下划线边框
- 支持图标的列表控件
- 在列表控件中显示单选按钮
- 列表控件折行显示列表项
- 使用图片制作绚丽按钮
- 实现按钮关键字描红
- 忙碌的按钮控件
- 实现透明效果的表格控件
- 在表格中显示工作进度百分比
- 在表格中显示图片
- 鼠标经过时按钮放大效果
- 迟到的登录按钮
- 焦点按钮的缩放
- 标签文本的跑马灯特效
- 延迟生效的按钮
- 动态加载表格数据
- 石英钟控件
- IP 输入文本框控件
- 日历控件
- 平移面板控件
- 背景图面板控件

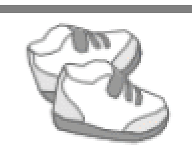
操作办公文档

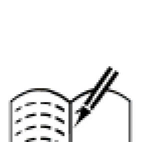
- 把文件文件导入到 word 中
- 浏览本地 word 文件
- 将员工表插入到 word 文档中
- 将员工表插入到 word 简历
- 将 word 文档保存为 HTML 格式
- 将员工信息保存到 Excel 表中
- 通过 Excel 公式计算出商品表中总售价
- 将数据表中内容写入到 Excel
- 将 Excel 表内容保存到数据库
- 将 Excel 转换为 HTML 格式
- 应用 iText 组件生成 PDF
- 在窗体中显示 PDF 文件
- 应用 PDF Renderer 组件实现放大 PDF 文件
- 应用 PDF Renderer 组件实现缩小 PDF 文件
- 应用 PDF Renderer 组件实现抓手功能
- 全屏显示 PDF 文件

文件、文件夹与系统

- 单表单元素上传文件到数据库
- 多表单元素上传文件到数据库
- 上传文件到服务器
- 限制文件大小的文件上传
- 遍历指定目录下的所有文件
- 获取驱动器信息
- 遍历指定驱动器
- 访问类路径上的资源文件
- 获取文件信息
- 查看文件是否存在
- 重命名文件
- 对文件夹创建、删除的操作
- 使用 Java 的输入输出流从文本文件中读取注册服务条款
- 使用 Java 的输入输出流实现永久计数器
- 通过文本文件向数据库传递数据
- 读取属性文件
- 复制文件夹
- 文件下载
- 使用 JSP 生成 XML 文档
- 使用 DOM 读取 XML 文件
- 使用 SAX 读取 XML 文件
- 修改文件属性
- 显示指定类型的文件
- 以树结构显示文件路径
- 查找替换文本文件内容
- 支持图片预览的文件选择对话框
- 设置 Windows 的文件属性
- 文件批量重命名
- 快速批量移动文件
- 删除磁盘所有.tmp 临时文件
- 提取数据库内容到文件
- 提取文本文件的内容到 MySQL 数据库
- 将图片文件保存到 SQL Server 数据库
- 显示数据库中的图片信息
- 提取技术网站数据到文件夹
- 读取文件路径到数据库
- 在数据库中建立磁盘文件索引
- 窗体动态加载磁盘文件
- 删除文件夹中所有文件
- 创建磁盘索引文件
- 快速全盘查找文件
- 获取磁盘所有文本文件
- 网络文件夹备份
- 键盘录入内容保存到文本文件





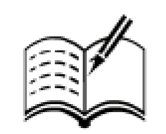


- 将数组写入到文件中并逆序输出
- 利用 StringBuffer 避免文件的多次写入
- 合并多个 txt 文件
- 实现文件简单加密与解密
- 对大文件实现分割处理
- 将分割后的文件重新合并
- 读取属性文件的单个属性值
- 向属性文件中添加信息
- 在复制文件时使用进度条
- 从 XML 文件中读取数据
- 读取 Jar 文件属性
- 电子通讯录
- 批量复制指定扩展名的文件
- 计数器小程序
- 将某文件夹中的文件进行分类存储
- 利用 StreamTokenizer 统计文件的字符数
- 在指定目录下搜索文件
- 序列化与反序列化对象
- 文件锁定
- 投票统计
- 压缩所有文本文件
- 压缩包解压到指定文件夹
- 压缩所有子文件夹
- 深层文件夹压缩包的释放
- 解决压缩包中文乱码
- Apache 实现文件解压缩
- 把窗体压缩成 ZIP 文件
- 解压缩 Java 对象
- 文件压缩为 RAR 文档
- 解压缩 RAR 压缩包
- 文件分卷压缩
- 为 RAR 压缩包添加注释
- 获取压缩包详细文件列表
- 从 RAR 压缩包中删除文件
- 在压缩文件中查找字符串
- 重命名 RAR 压缩包中的文件
- 创建自解压 RAR 压缩包
- 设置 RAR 压缩包密码
- 以压缩格式传输网络数据
- 压缩远程文件夹
- 压缩存储网页
- JSP 批量文件上传
- Struts 的文件上传
- Spring 的文件上传
- 从 FTP 下载文件
- 文件列表维护
- 文件在线压缩与解压缩
- 判断远程文件是否存在
- 通过文本文件实现网站计数器
- JSP 生成 XML 文件

数据库的操作

- 通过 JDBC-ODBC 桥连接 SQL Server 数据库
- 通过 JDBC 连接 SQL Server 数据库
- 通过 Tomcat 连接池连接 SQL Server 数据库
- 通过 WebLogic 连接池连接 SQL Server 数据库
- 应用 Hibernate 连接 SQL Server 数据库
- 通过 JDBC-ODBC 桥连接 Access 数据库
- 应用 Hibernate 连接 Access 数据库
- 通过 JDBC 连接 MySQL 数据库
- 通过 Tomcat 连接池连接 MySQL 数据库
- 应用 Hibernate 连接 MySQL 数据库
- 通过 JDBC 连接 Oracle 数据库
- 应用 Hibernate 连接
- 利用 SQL 语句实现分页
- 利用结果集进行分页
- 转到指定页的分页
- 具有页码跳转功能的分页
- 分栏显示
- 分类、分栏显示
- 对超长文本数据进行分页显示
- 单条数据录入
- 批量数据插入
- 插入用户登录日志信息
- 更新指定记录
- 批量更新
- 商品价格调整
- 修改密码
- 找回密码
- 动态创建 SQL Server 数据库
- 动态创建 SQL Server 数据表和字段
- 动态创建 MySQL 数据库
- 列举 SQL Server 数据库中的数据表
- 列举 MySQL 数据库中的数据表
- 查看数据表结构
- 在线维护投票数据库
- 通过 JDBC 获取插入记录的自动编号
- 通过 Hibernate 获取插入记录的自动编号
- 在线删除指定的一个数据表
- 在线删除多个指定的数据表
- 在线删除指定数据表中的指定索引
- 清空指定数据表中的所有数据
- 快速清空指定数据表中的所有记录
- 批量清空数据表中的数据
- 生成 SQL 数据库脚本
- 恢复 SQL 数据库脚本
- 删除指定记录
- 批量删除数据
- 删除数据前给予提示
- 获取从数据库里删除的记录数
- 生成有规律的编号
- 生成无规律的编号
- SQL Server 数据备份
- SQL Server 数据恢复
- 动态附加数据库
- 应用 JDBC 事务
- Hibernate 中应用事务
- 对数据进行降序查询
- 对数据进行多条件排序查询
- 对统计结果进行排序





- 查询 SQLServer 数据表中前 3 条数据
- 查询 SQLServer 数据库后 3 条数据
- 查询 MySQL 数据库中的前 3 条数据
- 查询 MySQL 数据库中后 3 条数据
- 按照字母顺序对留学生表进行排序
- 按姓氏笔画排序
- 将汉字按音序排序
- 按列的编号排序
- 从表中随机返回记录
- 使用 GROUP BY 函数实现对数据的分组统计
- 使用 GROUP BY 函数实现多表分组统计
- 利用 SUM 函数实现数据汇总
- 利用 AVG 函数实现计算平均值
- 利用 MIN 函数求数据表中的最小数据
- 利用 MAX 函数求数据表中的最大值
- 利用 COUNT 函数求销售额大于某值的图书种类
- 查询编程词典 6 月的销售量
- 查询与张静同一天入司的员工信息
- 使用 IN 谓词查询某几个时间的数据
- 日期查询中避免千年虫问题
- 在查询结果中不显示重复记录
- 使用 NOT 查询不满足条件的记录
- 使用 between 进行区间查询
- 列出销量表中重复记录和记录条数
- 使用关系运算符查询某一时间段数据
- 计算两个日期之间的月份数
- 格式化金额
- 在查询语句中过滤掉字符串中的空格
- 通过 JDBC-ODBC 桥连接 SQL Server2000
- 通过 JDBC 连接 SQLServer2000 数据库
- JDBC 连接 SQLServer2005 数据库
- JDBC 技术连接 Oracle 数据库
- JDBC 连接 JavaDB 数据库
- 列举 SQLServer 数据库下的数据表
- 列举 MySQL 数据库下的数据表
- 动态维护投票数据库
- MySQL 数据备份
- MySQL 数据恢复
- 获取 SQLServer 数据表字段的描述信息
- 将员工信息添加到数据表
- 添加数据时使用数据验证
- 在插入数据时过滤掉危险字符
- 将用户选择爱好以字符串形式保存到数据库
- 将数据从一张表复制到另张表
- 使用 UNION ALL 语句批量插入数据
- 在删除数据时给出提示信息
- 将数据表清空
- 字符串大小写转换
- 使用 JDBC 连接常用数据库
- JDBC 实现简单的搜索引擎
- 使用 JDBC 操作数据库
- 使用 JDBC 查看数据表结构
- 用户密码管理
- 使用 JDBC 批量处理数据
- 在 JDBC 中使用存储过程
- 使用 Tomcat 连接池
- 使用 dbcp 连接池
- 使用 c3p0 连接池
- MySQL 数据库的备份和恢复
- MySQL 数据库分页
- SQL Server 数据库备份与恢复
- SQL Server 数据库附加与分离
- SQL Server 数据库分页
- Hibernate 实现分页
- 将数据表导出到 XML 文件中
- 从 XML 文件导入数据表
- 在 MySQL 数据库增加新用户权限
- 使用事务提高数据库操作的安全性
- 在数据库中添加或读取文件数据
- 利用触发器记录系统登录日志
- 查询数值型数据
- 查询字符串
- 查询日期型数据
- 查询逻辑型数据
- 查询非空数据
- 查询文本框中指定的字符串
- 查询下拉列表框中指定的数值数据
- 查询下拉列表框中的日期数据
- 将表单元素中的内容作为字段、运算符和内容进行查询
- 利用变量查询字符串
- 利用变量查询数值型数据
- 查询前 5 名数据
- 查询后 5 名数据
- 取出数据统计结果前 3 名数据
- 查询指定 SQL Server 数据库中的日期型数据
- 查询指定 Access 数据库中的日期型数据
- 查询指定时间段的数据
- 按月查询数据
- 查询大于指定条件的数据
- 查询时不显示重复记录
- NOT 与谓词进行组合条件的查询
- 列出数据中的重复记录和记录条数
- 单列数据分组统计
- 多列数据分组统计
- 多表分组统计
- 利用聚集函数 SUM 对学生成绩




Note

- 利用聚集函数 SUM 对学生成绩进行汇总
- 利用聚集函数 AVG 求某班学生的平均成绩
- 利用聚集函数 MIN 求销售额最少的商品
- 利用聚集函数 MAX 求月销售额完成最多的员工
- 利用聚集函数 COUNT 求日销售额大于某值的图书种类数
- 利用 FROM 子句进行多表查询
- 使用表的别名
- 合并多个结果集
- 简单的嵌套查询
- 复杂的嵌套查询
- 用子查询作派生的表
- 用子查询作表达式
- 用子查询关联数据
- 多表联合查询
- 对联合查询后的结果进行排序
- 条件联合查询
- 简单内连接查询
- 复杂内连接查询
- 自连接
- LEFT OUTER JOIN 查询
- RIGHT OUTER JOIN 查询
- 使用外连接进行多表联合查询
- 利用 IN 谓词限定查询范围
- 用 IN 查询表中的记录信息
- 由 IN 引入的关联子查询
- 静态交叉表
- 动态交叉表
- 对查询结果进行格式化（四舍五入）
- 在查询中使用字符串函数
- 在查询中使用日期函数
- 利用 HAVING 语句过滤分组数据
- 复杂条件查询

数据库的高级应用

- 创建视图
- 使用视图过滤不想要的数据
- 使用视图与计算数据
- 使用视图重新格式化检索出来的数据
- 获取数据库中的全部用户视图
- 修改视图
- 删除视图
- 视图的应用
- 创建存储过程
- 调用存储过程实现用户身份验证
- 应用储存过程添加数据
- 调用加密存储过程
- 获取数据库中所有存储过程
- 修改存储过程
- 删除存储过程
- 应用存储过程实现数据分页
- 创建触发器
- 应用触发器自动插入回复记录
- 应用触发器添加日志信息
- 在删除成绩表时将学生表中数据删除
- 在程序中调用 UPDATE 触发器
- 获取数据库中的触发器名称
- 创建带有触发条件的触发器
- 使用批处理删除数据
- 使用批处理提升部门员工工资
- 将教师表中数据全部添加到选课表
- 在批处理中使用事务

实用的 JavaScript 函数

- 小写金额转换为大写金额
- 处理字符串中的空格
- 验证输入的日期格式是否正确
- 检查表单元素是否为空
- 验证 E-mail 是否正确
- 通过正则表达式验证电话号码
- 验证输入的字符串是否为汉字
- 验证身份证号码
- 客户端验证用户名和密码
- 验证网址是否合法
- 验证数量和金额
- 限制输入字符串的长度
- 显示长日期格式的系统日期
- 倒计时
- 实时显示系统时间
- 特殊日期提示

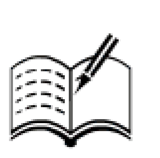
数据查询

- 将子查询作为表达式
- 用子查询作为派生表
- 通过子查询关联数据
- 使用 in 谓词限定查询范围
- 使用 NOT IN 子查询实现差集运算
- 使用 NOT IN 语句实现反向查询
- 返回笛卡尔乘积
- 比较运算符引入子查询
- 在子查询中使用聚合函数
- 在删除数据时使用子查询
- 查询平均成绩在 85 分以上的学生信息
- 查询本科部门经理月收入情况
- 在嵌套中使用 exists 关键字
- 动态指定查询条件
- 使用 UNION 运算符使学生档案归档
- 内联接获取指定课程的教师信息
- 左外连接查询员工信息
- 右外连接查询员工信息
- 多表外连接查询
- 完全连接查询
- 在查询中使用 patindex 函数进行模糊查询
- 对查询结果进行格式化
- 在查询中使用字符串函数
- 在查询中使用 ALL 谓词
- 在查询中使用 ANY 谓词
- 使用 UNION 运算符消除重复的行
- 使用 UNION ALL 运算符保留重复的行
- 计算商品销售额所占的百分比

Servlet 技术

- 将 HTML 元素嵌入到 Servlet
- 在 Servlet 中实现页面转发的操作
- 在 Servlet 中获取当前页的绝对路径





- 在 Servlet 中对 Cookie 的操作
- 利用 JavaBean 由 Servlet 向 JSP 传递数据
- 在 Servlet 中使用 JDBC- ODBC 桥访问数据库
- 在 Servlet 中使用 JDBC 访问数据库
- 使用 Servlet 访问数据库连接池
- 使用过滤器验证用户身份
- 使用过滤器进行网站流量统计
- 使用过滤器对响应页面中的敏感字符进行过滤
- 使用监听器查看在线用户
- 利用监听器使服务器端免登录
- 过滤非法文字
- 编码过滤器解决中文问题
- 过滤器验证用户
- 过滤器分析流量
- 使用过滤器禁止浏览器缓存页面
- 监听在线用户
- 监听器实现免登录

JavaBean 技术

- 连接数据库的方法
- 数据查询的方法
- 带参数的数据查询
- 数据增加的方法
- 数据修改的方法
- 数据删除的方法
- 数据库分页的方法
- 对结果集进行分页的方法
- 关闭数据库的方法
- 数据库事务处理的方法
- 调用数据库存储过程的方法
- 附加数据库的方法
- 备份数据库的方法
- 还原数据库的方法
- 自动获得汉字的拼音简码
- 转换输入文本中的回车和空格
- 小写金额转换为大写金额
- 判断字符串是否以指定字符开头
- 计算字符串的实际长度
- 字符串截取
- 字符串转换成数组
- 检查字符是否有英文字母
- 小写字母转换为大写字母
- 大写字母转换为小写字母
- 把数组转换成字符串
- 将整型数据格式化为指定长度的字符串
- 把一个长数字分位显示
- 过滤输入字符串中的危险符号
- 判断是否为当前时间的方法
- 判断用户输入的是否是数字的方法
- 对输入数据中的 HTML 字符进行转换的方法
- 过滤字符串中的空格与 null 值的方法
- 对 SQL 语句中输入的空值进行处理的方法
- 将整型值转换为字符型的方法
- 判断用户输入的是否为有效 id 值的方法
- 获取年份的方法
- 获取月份的方法
- 获取日的方法
- 显示指定格式的日期的方法
- 显示指定格式的时间的方法
- 显示完整日期时间的方法
- 对字符串进行 GBK 编码
- 对字符串进行 ISO-8859-1 编码
- 随机产生指定位数的验证码
- 生成指定位数的随机字符串
- 用户登录模块
- 带验证码的用户登录模块
- 带识别状态的用户登录模块
- 输出提示页面的方法
- 输出分页导航的方法
- 版权信息生成的方法
- 生成柱形图
- 生成折线图
- 生成饼状图
- 实现进度条
- 弹出提示对话框并重定向网页
- 打开指定大小的新窗口并居中显示

AJAX 技术

- Ajax 无刷新分页
- Ajax 实现聊天室
- Ajax 验证用户名是否被注册
- Ajax 刷新 DIV 内容
- Ajax 级联选择框
- Ajax 实现带进度条的文件上传

在线统计

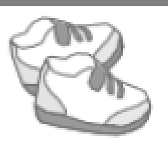
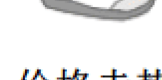
- 通过 Application 对象实现网站计数器
- 网站图形计数器
- 记录用户 IP 地址的计数器
- 只对新用户计数的计数器
- 统计用户在某一页停留的时间
- 统计用户在站点停留的时间
- 判断用户是否在线
- 实时统计在线人数
- 统计日访问量
- 利用柱形图统计分析网站访问量

网络通信

- 发送电子邮件
- 发送 HTML 格式邮件
- 带附件的邮件发送程序
- 邮件群发
- Spring 利用 Web Service 发送手机短信
- 利用短信猫发送手机短信
- 实现邮件发送
- 实现邮件接收
- 发送带附件的邮件
- 接收带附件的邮件
- 显示 POP3 未读邮件和已读邮件
- IP 地址转换成整数
- 获取本地天气预报

信息提取与图表分析

- 远程获取其他网页信息
- 网站访问量显示图表
- 投票结果显示图表
- 利用折线图分析多种商品的


Note

价格走势
- 利用折线图分析某种商品的价格走势
- 年销售额及利润图表分析

行业应用
- 禁止重复投票的在线投票系统
- 每个 IP 一个月只能投票一次的投票系统
- 一般用户注册
- 带检测用户名的用户注册
- 分步用户注册
- 论坛
- 购物车
- Application 形式的聊天室
- 聊天室（私聊）
- 数据库形式的聊天室（私聊）
- 简易万年历
- 带有备忘录的万年历

网站策略与安全
- 通过邮箱激活注册用户
- 越过表单限制漏洞
- 文件上传漏洞
- 防止 SQL 注入式攻击
- 获取客户端信息
- 防止网站文件盗链下载
- 禁止网页刷新
- 禁止复制和另存网页内容
- 防止页面重复提交
- 获取指定网页源代码并盗取数据
- 隐藏 JSP 网址扩展名
- 数据加密
- MD5 加密

安全技术
- 确定对方的 IP 地址
- 获取客户端 TCP/IP 端口的方法
- 替换输入字符串中的危险字符
- 禁止用户输入危险字符
- 用户安全登录
- 带验证码的用户登录模块
- 防止用户直接输入地址访问 JSP 文件
- 禁止复制网页内容
- 禁止网页被另存为
- 屏蔽 IE 主菜单
- 屏蔽键盘相关事件
- 屏蔽鼠标右键
- 对登录密码进行加密
- 字符串加密

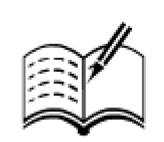
实用工具
- 在线查询 IP 地理位置
- 手机号码归属地查询
- 工行在线支付
- 支付宝的在线支付
- 快钱在线支付
- 在线文本编辑器
- 网页拾色器
- 在线验证 18 位身份证
- 在线汉字转拼音
- 在线万年历
- 进制转换工具

高级应用开发
- 自动选择语言跳转
- JSP 防刷计数器
- 用 JSP 操作 XML 实现留言板
- 网站支持 RSS 订阅
- JSP 系统流量分析
- 用 JSP 生成 WEB 静态网页

图形图像与多媒体
- 别致的图形计数器
- 预览并上传图片
- 分页浏览图像
- 生成中文验证码
- 生成关键字验证码
- 生成隐藏的验证码
- 生成带背景和雪花的验证码
- 生成带有干扰线的验证码
- 为图像添加文字水印和图片水印
- 制作相册幻灯片
- 在线连续播放音乐
- 在线播放 FLV 视频
- 同步显示 LRC 歌词
- JSP 生成条形码
- 通过下拉列表框选择头像
- 从网页对话框中选择头像
- 通过滑动鼠标放大或缩小图片
- 随机显示图片
- 幻灯片式图片播放
- 浮动广告
- 插入 Flash 动画
- 插入背景透明的 Flash
- 在线播放 MP3 歌曲列表
- MP3 文件下载
- 自制视频播放器
- 在线影片欣赏

报表打印
- 利用 JavaScript 调用 IE 自身的打印功能实现打印
- 利用 WebBrowser 打印
- 将页面中的客户列表导出到 Word 并打印
- 利用 Word 自动打印指定格式的会议记录
- 利用 Excel 打印工资报表
- 将 Web 页面中的数据导出到 Excel 并自动打印
- 打印库存明细表
- 打印库存盘点报表
- 打印库存汇总报表
- 打印指定条件的库存报表
- 打印汇款单
- 打印信封
- 利用柱形图分析报表
- 利用饼形图分析报表
- 利用折线图分析报表
- 利用区域图分析报表
- 导出报表到 Excel 表格
- 导出报表为 PDF 文档
- 实现打印报表功能
- 实现打印预览功能
- 用 JSP 实现 Word 打印
- JSP+CSS 打印简单的数据报表
- JSP 套打印快递单
- JSP 生成便于打印的网页
- 将数据库中的数据写入到 Excel
- 将数据库中的数据写入到 Word





第 2 部分　Java 模块资源库
（17 个经典模块分析）

模块 1　数码照片管理模块

- 模块概述
- 效果预览
- 关键技术——捕获树的选中节点事件
- 关键技术——浏览方式切换技术
- 关键技术——随意选取照片技术
- 关键技术——图片缩放与内存溢出
- 关键技术——工具提示回行显示技术
- 实现对相册树的维护
- 实现添加照片的功能
- 实现修改照片信息的功能
- 实现删除照片的功能
- 实现全屏查看照片功能
- 实现浏览方式的切换
- 实现查找照片功能 1
- 实现查找照片功能 2
- 实现图片播放器
- 保存选中图片到指定路径

模块 2　FTP 文件管理模块

- FTP 文件管理模块概述
- 关键技术
- 实现 FTP 站点管理功能
- 实现登录面板
- 实现本地资源管理
- 实现 FTP 资源管理
- 实现队列管理

模块 3　电子地图

- 模块概述
- 关键技术
- 实现地图处理器类
- 实现用来绘制地图的标签组件
- 实现操作地图功能
- 实现维护标记功能
- 实现搜索标记功能

模块 4　网络五子棋游戏

- 五子棋模块概述
- 关键技术
- 实现登录界面
- 编写游戏主窗体
- 编写下棋面板
- 编写棋盘面板
- 实现游戏规则算法
- 编写棋盘模型
- 编写联机通讯类

模块 5　远程协助模块

- 远程协助模块介绍
- 关键技术
- 联系人管理
- 创建网络服务器
- 编写远程连接面板
- 启动 RMI 远程方法服务
- 实现远程监控界面
- 实现系统托盘

模块 6　软件注册模块

- 软件注册模块概述
- 关键技术
- 软件注册导航窗体的实现
- 软件注册窗体的实现
- 注册机的实现

模块 7　多媒体播放器模块

- 模块概述
- 关键技术
- 实现播放媒体文件
- 实现播放控制
- 播放列表维护
- 播放控制
- 创建最近播放列表
- 实现自动检索系统中媒体文件

模块 8　决策分析模块

- 模块概述
- 数据接口
- 关键技术
- 实现过程

模块 9　网站留言簿

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 数据表结构
- 创建 Hibernate 配置文件
- 创建实体类及映射文件——创建实体类
- 创建实体类及映射文件——创建映射文件
- 业务处理逻辑类——对留言及回复信息操作的 Topic 类 1
- 业务处理逻辑类——对留言及回复信息操作的 Topic 类 2
- 业务处理逻辑类——对管理员操作的 Manager 类
- 创建公共类——ChStr 类和 GetTime 类
- 创建公共类——style 样式文件和 check 类
- 添加留言信息 1
- 添加留言信息 2
- 显示留言信息 1
- 显示留言信息 2
- 显示留言信息 3
- 回复留言 1
- 回复留言 2
- 删除留言
- 用户登录页面
- 公共页
- 调试、发布与运行

模块 10　桌面精灵

- 模块概述





Note

- 关键技术
- 实现滚动字幕
- 实现支持农历的万年历
- 实现维护记录功能
- 实现搜索记录功能

模块 11　短信发送模块

- 短信发送模块概述
- 关键技术
- 数据库设计
- 设置并连接短信猫
- 读取短信
- 发送短信
- 发信箱的实现
- 联系人管理

模块 12　数据分页

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 开发过程
- 发布与运行

模块 13　电子阅读器模块

- 电子阅读模块概述
- 关键技术
- 实现主窗体
- PDF 文档读取的实现
- 缩位图的实现
- 书签的实现
- 全屏显示 PDF 文档

模块 14　复杂条件查询

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 数据表结构
- 创建 Hibernate 配置文件
- 创建实体类及映射文件——创建实体类 1
- 创建实体类及映射文件——创建实体类 2
- 创建实体类及映射文件——创建映射文件
- 业务处理逻辑类 1
- 业务处理逻辑类 2
- 创建公共类——style 样式文件和 ChStr 类
- 创建公共类——function.js
- 创建公共类——menu.JS
- 创建公共类——onclock.JS 文件和 web.xml 文件
- 查询并显示全部数据
- 指定条件查询并显示数据 1
- 指定条件查询并显示数据 2
- 系统首页
- 调试、发布与运行

模块 15　权限管理

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 数据表结构
- 创建 Hibernate 配置文件
- 创建实体类及映射文件——创建实体类 1
- 创建实体类及映射文件——创建实体类 2
- 创建实体类及映射文件——创建映射文件
- 业务处理逻辑类 1
- 业务处理逻辑类 2
- 业务处理逻辑类 3
- 创建公共类
- 添加管理员信息
- 显示管理员列表
- 修改管理员权限
- 删除管理员信息
- 用户登录
- 发布与运行

模块 16　网页浏览器模块

- 浏览器模块概述
- 关键技术
- 实现菜单栏
- 工具栏的实现
- 地址栏的实现
- 实现浏览器选项卡
- 实现收藏夹

模块 17　企业内部邮件管理

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 数据表结构
- 创建配置文件、实体类和映射文件 1
- 创建配置文件、实体类和映射文件 2
- 创建配置文件、实体类和映射文件 3
- 创建配置文件、实体类和映射文件 4
- 创建配置文件、实体类和映射文件 5
- 用户注册
- 发送邮件
- 接收邮件
- 查看邮件
- 删除邮件
- 用户登录页面

第 3 部分　Java 项目资源库
（17 个项目开发案例）

项目 1　Validator 验证框架

- 概述与开发环境
- 实例运行结果
- 设计与分析
- 技术要点
- 开发过程



调试、发布与运行

项目 2　基于 Struts 开发的网上企业办公自动化

概述与开发环境
需求分析
数据库设计
前期准备
网站总体设计
公共类的编写
登录模块
自定义标签的开发
收/发文管理模块
会议管理模块
公告管理模块
人力资源管理模块
文档管理
资产管理模块
内部邮件管理模块
意见箱模块
退出模块
疑难解答
公共模块

项目 3　基于 Struts 与 IBatis 开发的图书管理系统

概述与开发环境
需求分析
系统设计
数据库设计
网站总体设计
公共类的编写
IBatis 设计模式的介绍
配置 Struts
ActionForm 类的编写及配置
对数据表操作持久类的编写
图书管理系统总体架构
管理员功能模块
图书管理功能模块
图书借还管理功能模块
图书设置模块
图书销售模块
图表分析模块
公共模块

疑难问题分析与解决

项目 4　进销存管理系统

开发背景和系统分析
系统设计
数据库设计
主窗体设计
公共模块设计
基础信息模块设计
进货管理模块设计
查询统计模块设计
库存管理模块设计
系统打包发布
开发技巧与难点分析
使用 PowerDesigner 逆向生成数据库 E-R 图

项目 5　基于Struts与Hibernate开发的新闻网络中心

概述、开发环境与需求分析
系统设计
数据库设计
网站总体设计
公共类的编写
Hibernate 数据库配置文件
ActionForm 类的编写及配置
对数据表操作持久类的编写
网站后台主要功能设计
网站前台主要功能设计
公共模块

项目 6　蜀玉网络购物商城

概述与系统分析
总体设计
系统设计
技术准备
系统架构设计
JavaBean 的设计
会员管理模块
网站主页设计
会员资料修改模块
购物车模块
商品销售排行模块
网站后台主要功能模块设计
退出模块

疑难问题分析

项目 7　博客网站

概述与系统分析
总体设计
系统设计
技术准备
系统构架设计
公共类设计
网站前台主要功能设计
网站后台主要功能模块设计
疑难问题分析
程序调试与错误处理

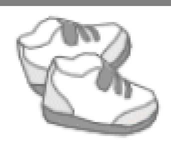
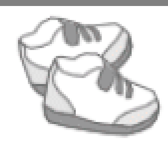
项目 8　企业内部通信系统

开发背景和系统分析
系统设计
主窗体设计
公共模块设计
系统托盘模块设计
系统工具模块设计
用户管理模块设计
通信模块设计
开发技巧和 JDK 6 新增的系统托盘

项目 9　网络购物中心

概述与系统分析
项目规划
系统设计
技术准备
网站的总体架构
配置 Struts
前台系统架构设计
前台网站前台首页设计
前台用户管理模块
前台购物车模块
前台商品信息查询模块
前台商城公告模块
后台系统架构设计
后台网站后台首页设计
后台管理员身份验证模块
后台商品设置模块
后台订单设置模块
后台公告设置模块

Note

Note


- 后台会员设置模块
- 后台管理员设置模块
- 后台友情链接设置模块
- 后台退出后台模块
- 大类别商品管理模块公共类
- 订单管理模块公共类
- 公告管理模块公共类
- 管理员管理模块公共类
- 商品管理模块公共类
- 小类别商品管理模块公共类
- 用户管理模块公共类
- 友情链接模块公共类
- 疑难问题分析
- 程序调试与错误处理

项目 10　企业人事管理系统
- 开发背景和系统分析
- 系统设计
- 数据库设计
- 主窗体设计
- 公共模块设计
- 人事管理模块设计
- 待遇管理模块设计
- 系统维护模块设计
- 开发技巧与难点分析
- Hibernate 关联关系的建立方法

项目 11　销售管理系统
- 概述与系统分析
- 总体设计
- 系统设计
- 技术准备
- 网站整体设计
- 工具类的代码实现过程
- 对员工操作模块设计
- 员工注册和员工权限的实现
- 分页的实现
- 修改员工信息
- 公司信息修改与查询
- 公共模块代码与公共类
- 首页模块设计
- 基础信息设置模块设计
- 单位类型与计量单位模块设计
- 支付方式与银行名称模块设计
- 企业资信与商品类别模块设计
- 公共类与公共模块代码
- 基础资料模块设计
- 查看商品详细信息模块设计
- 公共类与公共模块代码
- 业务管理模块设计
- 公共类与公共模块代码
- 信息查询模块设计
- 辅助工具模块设计
- 疑难问题解析
- 程序调试与错误处理

项目 12　医药进销存管理系统
- 概述与系统分析
- 总体设计
- 系统设计
- 技术准备
- 系统总体架构设计
- 系统公共类设计
- Java 实体类及 Hibernate 映射文件的设计
- 系统功能模块设计
- 系统主界面设计
- 设计药品基本情况模块
- 设计客户基本情况模块
- 设计供应商基本情况模块
- 设计药品采购模块
- 设计药品销售模块
- 设计库存盘点模块
- 设计销售退货模块
- 设计客户回款模块
- 设计基本信息查询模块
- 设计入库明细查询模块
- 设计销售明细查询模块
- 设计回款信息查询模块
- 设计增加用户功能模块
- 设计用户维护功能模块
- 系统提示模块设计
- 疑难问题解析
- 系统常见错误处理

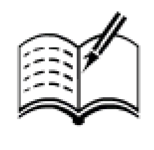
项目 13　图书馆管理系统
- 开发背景和需求分析
- 系统设计
- 数据库设计
- 公共模块设计
- 主窗体设计
- 登录模块设计
- 图书信息管理模块设计
- 图书借阅、归还模块设计
- 图书查询模块设计
- 开发技巧与难点分析
- 格式化的文本框

项目 14　销售管理系统
- 概述与系统分析
- 总体设计
- 系统设计
- 技术准备
- 系统架构设计
- 工具类的设计与实现
- 员工操作模块的设计与实现
- 基础信息设置模块的设计与实现
- 业务管理模块的设计与实现
- 疑难问题解析

项目 15　酒店管理系统
- 概述与系统分析
- 总体设计
- 系统设计
- 技术准备
- 系统架构设计
- 数据持久层设计
- 主窗体的格局设计
- 开台签单功能的设计与实现
- 自动结账功能的设计与实现
- 销售统计功能的设计与实现
- 人员管理功能的设计与实现
- 系统维护功能的设计与实现
- 系统安全功能的设计与实现
- 疑难问题分析

项目 16　企业快信
- 企业快信概述和系统分析
- 系统设计
- 数据库设计
- 主窗体设计
- 公共模块设计





- 资源管理模块设计
- 发送短信模块设计
- 发送邮件模块设计
- 系统设置模块设计
- 开发技巧与难点分析
- 使用短信猫和 Java Mail 组件

项目 17　通用固定资产及设备管理系统

- 概述与系统分析
- 总体设计
- 系统设计
- 技术准备
- 系统总体架构设计
- 公共类的设计
- 实体 Bean(CMP)和会话 Bean 的设计
- 编写配置文件
- 系统主要功能设计
- 资产/设备添加模块设计
- 资产/设备基本信息维护模块设计
- 资产/设备借出归还管理模块设计
- 资产/设备维修管理模块设计
- 资产/设备借出归还信息查询设计
- 用户管理模块设计
- 资产设备查询模块设计
- 资产/设备借出归还登记情况模块设计
- 资产设备信息浏览模块设计
- 资产设备折旧情况模块设计
- 疑难解析
- 数据表附录

第 4 部分　Android 任务资源库

（99 个编程实践任务）

Andriod 模拟器与工具

- 设置模拟器声音
- 查看模拟器存储状态
- 使用 DDMS 透视图查看 Android 文件系统结构
- 卸载搜狗拼音输入法

用户界面设计

- 简易的图片浏览器
- 开发自定义的 View 在窗体上绘制一只地鼠
- 应用相对布局实现显示软件更新提示界面
- 使用表格布局与线性布局实现分类工具栏

Android 提供的组件

- 实现我同意游戏条款
- 猜猜鸡蛋放在哪只鞋子里
- 实现带图标的 ListView 列表
- 实现仿 Windows 7 图片预览窗格效果

深入理解 Activity

- 实现启动和关闭 Activity
- 实现应用对话框主题的关于 Activity
- 实现根据输入的生日判断星座
- 实现带选择所在城市的用户注册界面
- 实现古诗欣赏程序
- 实现在一屏上显示的图片查看器

意图与广播

- 使用 Intent 查看通讯录信息
- 使用 Intent 修改通讯录信息
- 使用隐式 Intent 打开拨号界面
- 使用隐式 Intent 打开短信发送界面
- 接收短信后显示短信号码
- 接收短信后显示短信内容

用户资源访问

- 通过字符串资源显示游戏对白
- 使用颜色资源设置窗体的背景颜色
- 通过尺寸资源将文字逐个放大
- 使用数组资源和 ListView 显示联系人列表
- 使用 9-Patch 图片实现登录和退出按钮
- 实现自定义复选按钮的样式
- 应用样式资源改变文字的样式
- 应用主题资源给所有窗口添加背景
- 从 XML 文件中读取新闻信息
- 实现带子菜单的上下文菜单
- 创建一组只能单选的选项菜单
- 对选项菜单进行国际化
- 对 ListView 列表进行国际化

Android 事件处理

- 显示用户单击的按键
- 判断是否为系统按键
- 显示用户触摸时持续的时间
- 显示用户触摸屏幕位置
- 查看手势对应分值
- 使用手势输入数字

通知、对话框与警告

- 实现单击按钮时显示简单的消息提示
- 实现单击按钮时显示带图标的消息提示
- 弹出选择颜色的单选列表对话框
- 应用 AlertDialog 实现自定义的登录对话框
- 发送一个在状态栏上显示的通知
- 发送一个自定义声音提示的通知
- 设置连续响铃的闹钟
- 设置一个 BroadcastReceiver 闹钟

Android 程序调试

- 向 LogCat 视图中输出程序 Info 日志
- 将两个整数的和作为冗余日志输出到 LogCat 视图中
- 应用 throws 抛出类未发现异常
- 使用 throw 关键字在方法中抛出异常




Note

- 图形图像与动画
  - 绘制渐变色填充的圆形
  - 使用 ImageView 显示 SD 卡上的图片
  - 在屏幕上绘制小房子
  - 在屏幕上绘制彩色字符串
  - 绘制带描边的圆角矩形图片
  - 实现探照灯效果
  - 实现闪烁的星星
  - 实现来回捕食的小鱼
- 多媒体应用开发
  - 使用 MediaPlayer 和 SurfaceView 实现带音量控制的视频播放器
  - 实现控制是否播放按键音
  - 实现在拍摄相片上添加拍照日期
  - 实现在拍摄相片上添加边框
- Android 数据存储技术
  - 使用 SharedPreferences 在 Activity 间传递整数值
  - 使用 SharedPreferences 在 Activity 间传递布尔值
  - 显示内部存储文件位置的绝对路径
  - 获得当前内部存储中创建的全部文件名称列表
  - 判断获得的 SD 卡内容是否是文件夹
  - 显示文件和文件夹的创建时间
  - 使用列表显示数据表中全部数据
  - 使用列表逆序显示数据表中全部数据
- Content Provider 使用
  - 使用列表显示联系人 ID 和职务信息
  - 使用列表显示联系人 ID 和公司信息
- 线程与消息处理
  - 实现在日志窗口中每隔 1 秒显示一个文字
  - 开启新线程播放背景音乐
  - 实现在屏幕上来回移动的气球
  - 实现颜色不断变化的文字
- Service 应用
  - 应用启动 5 秒钟后停止 Service
  - 应用启动 10 秒钟后停止 Service
  - 视力保护程序
  - 查看当前运行服务信息
- 利用 OpenGL 实现 3D 图形
  - 绘制长方体
  - 绘制三棱锥
  - 绘制一个不断旋转的三棱锥
  - 绘制一个不断旋转的三棱锥
- Action Bar 使用
  - 显示自定义视图
  - 重新设置 Icon 图标
- 定位服务
  - 显示海拔信息
  - 显示方向信息
  - 显示成都交通图信息
  - 显示成都卫星图信息
- 网络编程及 Internet 应用
  - 实现通过 GET 请求发送中文参数
  - 使用 ImageView 显示从网络上获取的图片
  - 实现带刷新和缩放功能的网页浏览器
  - 获取天气预报

第 5 部分　Java 能力测试资源库
（616 道能力测试题目）

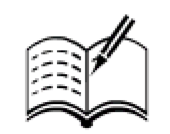
- Java 编程基础能力测试
  - Java 概述
  - 使用 IDE 继承开发工具
  - Java 语言基础知识
  - 流程控制
  - 数组应用
  - 面型对象编程
  - 字符串处理
  - 类的继承与多态特征
  - 其他类特性与异常处理
  - Swing 编程基础
  - 多线程编程
  - 事件处理的应用
  - 常用工具类
  - 常用集合类
  - 数据库编程应用
  - 输入输出流与文件
  - 网络程序设计
  - 表格控制的应用
  - 其他高级控件
  - 图形绘制技术
- 数学及逻辑思维能力测试
  - 基本测试
  - 进阶测试
  - 高级测试
- 编程英语能力测试
  - 英语基础能力测试
  - 英语进阶能力测试

第 6 部分　编程人生
（23 个 IT 励志故事）

- 励志故事
  - “盖茨第二”——马克・扎克伯格
  - 微型博客 Twitter——埃文・威廉姆斯
  - 缔造华人的硅谷传奇——

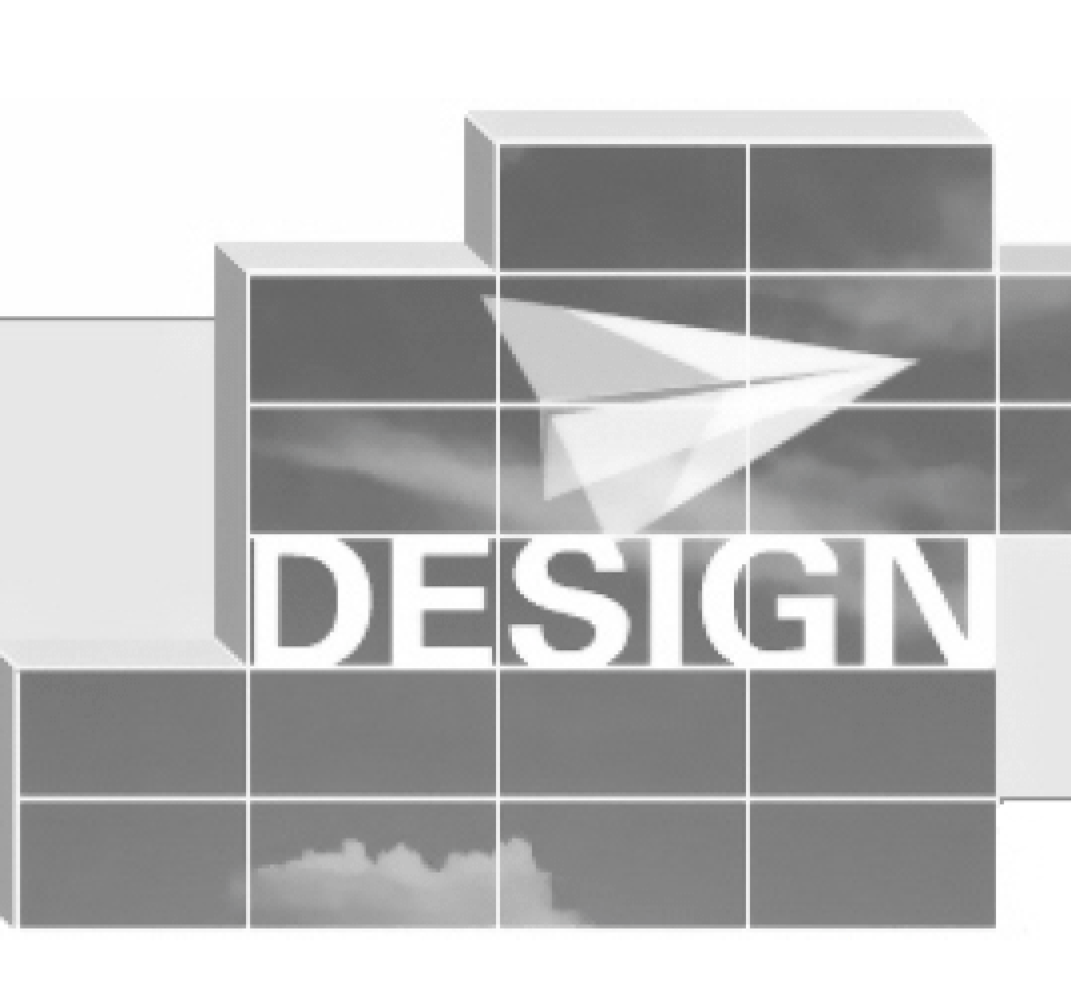


杨致远
- 玩出传奇一世界第一人称射击游戏之父——约翰·卡马克
- 因特网的点火人——马克·安德森
- 不可思议的传奇人生——“杀毒王”王江民
- 暴雪公司的领航者——迈克·莫汉
- IT“大王”——王志东
- 中国第一程序员——求伯君
- IT 风云人物——鲍岳桥
- 征途巨人——史玉柱
- 创造互联网搜索时代——拉里·佩奇和谢尔盖·布林
- 不断挑战自己的成功——徐少春
- 专注是通往成功的桥梁——陈天桥
- BEA 创始人——庄思浩
- 初中站长的创业故事——李兴平
- 软件业的华人教父——王嘉廉
- 点燃 JAVA 技术之火——詹姆斯·戈士林
- 使计算机成为生活的必需品——比尔·盖茨
- 中国通信设备行业的领跑者——任正非
- 知识改变命运、科技改变生活——李彦宏
- 为编程事业而奋斗终生——安德斯
- 让下载迅雷不及掩耳——邹胜龙



Note

第 1 篇



入门篇

- 第 1 章　Android 入门
- 第 2 章　搭建 Android 开发环境
- 第 3 章　认识 Android 模拟器
- 第 4 章　剖析 Android 程序
- 第 5 章　Android 常用组件的使用
- 第 6 章　掌握布局管理器
- 第 7 章　Android 程序调试与错误处理
- 第 8 章　Activity 的使用
- 第 9 章　使用 Intent 进行通信

第 1 章

Android 入门

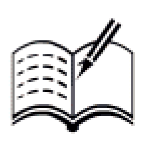
（视频讲解：22 分钟）

Android 是 Google 公司推出的一种基于 Linux 平台的开源操作系统，主要分为手机操作系统和平板电脑操作系统。本章将首先向读者介绍 Android 及其相关的特性、不同的版本，以及学好 Android 的方法等内容。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ Android 的体系结构
- ☐ Android 的最新版本
- ☐ 说出现在市场上常用的 5 种 Android 应用
- ☐ 访问 Android 应用的官方商店
- ☐ 描述 Android 的主要特性
- ☐ 描述 Android 4.3 中的新增特性
- ☐ 使用 Android API 文档





Note

1.1　Android 概述

Android 本义是指“机器人”，它是 Google 公司推出的一款开源操作系统，随着 Android 操作系统在手机和平板电脑市场的普及，Android 应用的需求势必会越来越大。本节将向大家简单介绍 Android，并介绍几种学习 Android 的方法。

1.1.1　Android 的定义

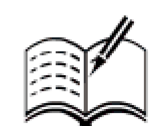
Android 是 Google（谷歌）于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，中文名为安卓（官方）。Android 系统早期由原名为“Android”的公司开发，该公司的创始人是 Andy Rubin，Google 在 2005 年收购“Android.Inc”后，聘用 Andy Rubin 作为 Android 产品的负责人，继续对 Android 系统开发运营。Android 系统采用了软件堆层（software stack，又名软件叠层）的架构，主要分为 3 部分，其中，底层 Linux 内核只提供基本功能，其他的应用软件则由各公司自行开发，部分程序以 Java 语言编写。Android 的体系结构如图 1.1 所示。



图 1.1　Android 的体系结构

图 1.1 是 Android 官网上给出的 Android 体系结构图，从该图中可以看出，Android 分为 4



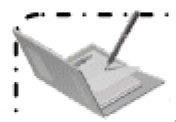


层，分别是 Linux 内核层、系统运行库层、应用框架层和应用层，下面分别对这 4 层及其功能进行介绍。

1. Linux 内核层（LINUX KERNEL）

Android 的核心系统服务是基于 Linux 内核的，如安全性、内存管理、进程管理、网络协议栈和驱动模型等都依赖于该内核。Linux 内核同时也作为硬件和软件栈之间的抽象层，而 Android 更多的是需要一些与移动设备相关的驱动程序，其主要驱动如下。

- ☑ Display Driver：显示驱动，基于 Linux 的帧缓冲驱动。
- ☑ Camera Driver：照相机驱动，基于 Linux 的 v412 驱动。
- ☑ Bluetooth Driver：蓝牙驱动，基于 IEEE 802.15.1 标准的无线传输技术。
- ☑ Flash Memory Driver：Flash 闪存驱动，基于 MTD 的 Flash 驱动程序。
- ☑ Binder(IPC) Driver：Android 的一个特殊的驱动程序，具有单独的设备节点，提供进程间通信的功能。
- ☑ USB Driver：USB 接口驱动。
- ☑ Keypad Driver：键盘驱动，作为输入设备的键盘驱动。
- ☑ WiFi Driver：基于 IEEE 802.11 标准的驱动程序。
- ☑ Audio Drivers：音频驱动，基于 ALSA（Advanced Linux Sound Architecture）的高级 Linux 声音体系驱动。
- ☑ Power Management：电源管理，如电池电量等。



说明：

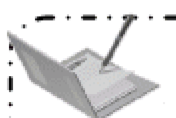
最新的 Android 4.3 版本基于 Linux 3.4 内核。

2. 系统运行库层（LIBRARIES）

系统运行库层主要提供 Android 程序运行时需要的一些类库，这些类库一般是使用 C/C++语言编写的，另外，该层还包含了 Android 运行库。系统运行库层中包含的主要库如下。

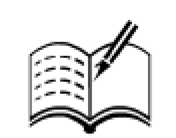
- ☑ libc：C 语言标准库，系统最底层的库，其通过 Linux 系统来调用。
- ☑ Surface Manager：主要管理多个应用程序同时执行时，各个程序之间的显示与存取，并且为多个应用程序提供了 2D 和 3D 涂层的无缝融合。
- ☑ SQLite：关系数据库。
- ☑ OpenGL|ES：3D 效果的支持。
- ☑ Media Framework：Android 系统多媒体库，该库支持多种常见格式的音频、视频的回放和录制，如 MPEG4、MP3、AAC、JPG 和 PNG 等。
- ☑ WebKit：Web 浏览器引擎。
- ☑ SGL：2D 图形引擎库。
- ☑ SSL：位于 TCP/IP 协议与各种应用层协议之间，为数据通信提供支持。
- ☑ FreeType：位图及矢量库。





说明：

系统运行库层中还包含了一个 Dalvik 虚拟机，该虚拟机相对于桌面系统和服务器系统运行的虚拟机而言，不需要很快的 CPU 计算速度和大量的内存空间，因此，它非常适合在移动终端上使用。



3．应用框架层（APPLICATION FRAMEWORK）

应用框架层是编写 Google 发布的核心应用时所使用的 API 框架，开发人员可以使用这些框架来开发自己的应用程序，这样可以简化程序开发的架构设计。Android 应用框架层提供的主要 API 框架如下。

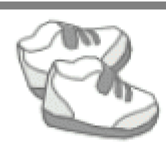
- ☑ Activity Manager：活动管理器，用来管理应用程序声明周期，并提供常用的导航退回功能。
- ☑ Window Manager：窗口管理器，用来管理所有的窗口程序。
- ☑ Content Providers：内容提供器，它可以让一个应用访问另一个应用的数据，或共享它们自己的数据。
- ☑ View System：视图管理器，用来构建应用程序，如列表、表格、文本框及按钮等。
- ☑ Notification Managere：通知管理器，用来设置在状态栏中显示的提示信息。
- ☑ Package Manager：包管理器，用来对 Android 系统内的程序进行管理。
- ☑ Telephony Manager：电话管理器，用来对联系人及通话记录等信息进行管理。
- ☑ Resource Manager：资源管理器，用来提供非代码资源的访问，如本地字符串、图形及布局文件等。
- ☑ Location Manager：位置管理器，用来提供使用者的当前位置等信息，如 GPRS 定位。
- ☑ XMPP Service：Service 服务。

4．应用层（APPLICATIONS）

应用层是用 Java 语言编写的运行在 Android 平台上的程序，如 Google 默认提供的 E-mail 客户端、SMS 短信、日历、地图及浏览器等程序。作为 Android 开发人员，通常需要做的就是编写在应用层上运行的应用程序，如大家所熟知的愤怒的小鸟、植物大战僵尸和微博客户端等程序。

1.1.2 Android 成功案例

为了更好地推广 Android，Google 和几十个手机相关企业建立了开放手机联盟（Open Handset Alliance），联盟成员包括摩托罗拉（Motorola）、HTC、SAMSUNG、LG、Intel、NVIDIA、SiRF、Skype、KUPA Map、MTK 以及中国移动在内的 34 家技术和无线应用的领军企业。而在实际应用中，Android 项目也已经有了很多成功的项目案例，如大家所熟知的美图秀秀、好联络等应用软件，以及愤怒的小鸟、植物大战僵尸等游戏，它们在 Android 操作系统上运行的效果分别如图 1.2～图 1.5 所示。




Note



图 1.2　Android 应用之美图秀秀



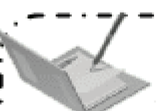
图 1.3　Android 应用之好联络



图 1.4　Android 应用之愤怒的小鸟



图 1.5　Android 应用之植物大战僵尸



说明：

（1）图 1.2～图 1.5 演示的是运行在 Android 4.3 虚拟机上的效果。

（2）摩托罗拉（Motorola）于 2011 年 8 月 15 日被 Google 收购，这也更进一步凸显了 Google 发展 Android 操作系统的决心。

1.1.3　Android 的版本

Android 用甜点作为系统版本的代号，该命名方法开始于 Android 1.5 版本，作为每个版本代表的甜点的尺寸越变越大，然后按照 26 个字母数排序：纸杯蛋糕、甜甜圈、松饼、冻酸奶、姜



饼、蜂巢、冰淇淋三明治、果冻豆……Android 迄今为止发布的主要版本及其发布时间如下。

1．Android 1.1

发布时间：发布于 2008 年 9 月。

2．Android 1.5

代号：Cupcake（纸杯蛋糕）。
发布时间：发布于 2009 年 4 月。

3．Android 1.6

代号：Donut（甜甜圈）。
发布时间：发布于 2009 年 9 月。

4．Android 2.0

代号：Éclair（松饼）。
发布时间：发布于 2009 年 10 月 26 日。

5．Android 2.1

代号：Éclair（松饼）。
发布时间：发布于 2009 年 10 月 26 日，Android 2.0 版本的升级以创纪录的速度放出。

6．Android 2.2

代号：Froyo（冻酸奶）。
发布时间：发布于 2010 年 5 月 20 日。

7．Android 2.3

代号：Gingerbread（姜饼）。
发布时间：发布于 2010 年 12 月 7 日。

8．Android 3.0

代号：Honeycomb（蜂巢）。
发布时间：发布于 2011 年 2 月 3 日。

9．Android 3.1

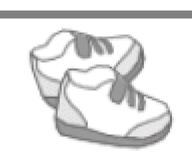
代号：Honeycomb（蜂巢）。
发布时间：发布于 2011 年 5 月 10 日。

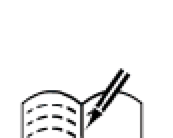
10．Android 3.2

代号：Honeycomb（蜂巢）。
发布时间：发布于 2011 年 7 月 13 日。

11．Android 4.0

代号：Ice Cream Sandwich（冰淇淋三明治）。
发布时间：发布于 2011 年 10 月 19 日。





12．Android 4.1

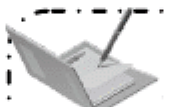
代号：Jelly Bean（果冻豆）。
发布时间：发布于 2012 年 6 月 28 日。

13．Android 4.2

Jelly Bean（果冻豆）。
发布时间：发布于 2012 年 10 月 30 日。

14．Android 4.3

Jelly Bean（果冻豆）。
发布时间：发布于 2013 年 7 月 25 日。



说明：

Android 3.0（蜂巢）之前的版本主要针对移动手机，Android 蜂巢版本系列（即 3.0、3.1 和 3.2 版本）主要针对平板电脑及上网本，而 Android 4.0 之后的版本将同时支持移动手机、平板电脑及上网本等终端。

1.1.4 Android 市场

Android 市场是 Google 公司为 Android 平台提供的在线应用商店，Android 平台用户可以在该市场中浏览、下载和购买第三方人员开发的应用程序。

对于开发人员，有两种挣钱的方式：一种方式是卖软件，开发人员可以获得该应用售价的 70%，其余 30%作为其他费用；另一种方式是加广告，将自己的软件定为免费软件，通过增加广告链接，靠点击率挣钱。

1.2 Android 特性

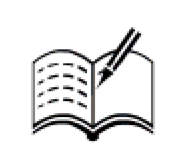
Android 作为一种开源操作系统，其在手机操作系统领域的市场占有率已经超过了 50%，是什么原因让 Android 操作系统如此受欢迎呢？本节将介绍 Android 的一些主要特性。

1.2.1 开放性

在优势方面，Android 平台首要的优势就是其开放性，开放的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于





消费者来讲，最大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

1.2.2　挣脱束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。自从 iPhone 上市，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈。

1.2.3　丰富的硬件

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步，甚至软件的兼容。例如你从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用，联系人等资料更是可以方便地转移。

1.2.4　开发商

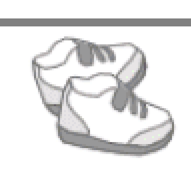
Android 平台提供给第三方开发商一个十分宽泛、自由的环境，因此不会受到各种条条框框的阻挠，可想而知，会有多少新颖别致的软件诞生，但也有其两面性，血腥、暴力、情色方面的程序和游戏如何控制正是留给 Android 的难题之一。

1.2.5　Google 应用

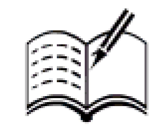
如今叱咤互联网的 Google 已经走过数十年历程，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

1.3　Android 4.3 新增特性

Android 4.3 系统的研发代号是 Jelly Bean（果冻豆），于北京时间 2013 年 7 月 25 日凌晨在 Google 新品发布会上发布。相比于 Android 4.2，Android 4.3 系统并未在用户界面上做出过多改变，保持了果冻豆（Jelly Bean）系列统一的 Holo 风格。Android 4.3 虽然没有加入颠覆性的新功能，但实际上在系统内部进行了一系列提升。根据最新的 AOSP 格式更新日志显示，Android 4.3



系统已悄然改进了超过 3.5 万项内容，大大增强了其安全性、易用性和拓展性。本节将对 Android 4.3 版本中新增的特性进行介绍。



Note

1.3.1 用户体验

果冻豆（Jelly Bean）系列系统在黄油项目（Project Butter）的帮助下，已引入“垂直同步定时”（Vsync Timing）、“三重缓冲”（Triple Buffering）、“减少的触摸延时”（Reduced Touchlatency）、“CPU 输入提振”（CPU input Boost）和“硬件加速的 2D 渲染”（Hardware-accelerated 2D Rendering）等技术，令安卓设备运行起来达到了前所未有的顺滑。而作为 Android 4.X 系列的收官之作，Android 4.3 系统再次增加了新的优化：对于图形性能，硬件加速 2D 渲染优化了流绘图命令；对于多线程处理，渲染也可以使用多个 CPU 内核的多线程执行某些任务；此外，新系统还对形状和文本的渲染进行了提升，并改进了窗口缓冲区的分配。所有这一切，都将会为用户带来一个全新的安卓体验，快速、流畅而灵敏。

1.3.2 多用户切换与受限账户

允许同一台设备拥有最多 8 个独立的用户空间，并且可以保持 3 个账户的活跃状态，并且优化了锁屏界面的用户切换速度，再多用户也无压力。

1.3.3 蓝牙

Android 4.3 系统正式支持低能耗蓝牙 4.0 技术（Bluetooth 4.0 Low Energy）。相较于 3.0 版本，蓝牙 4.0 拥有低功耗、3 毫秒低延迟和 AES-128 加密等特点，它将 3 种规格集一体，包括传统蓝牙技术、高速技术和低耗能技术。根据官方的数据，蓝牙 4.0 的峰值能耗约为 3.0 的一半，极低的运行和待机功耗使得一粒纽扣电池甚至可连续工作一年之久。考虑到安卓设备的全球占有率情况，低能耗的连接方式，将会促进可穿戴设备的普及，并加速物联网的建设。

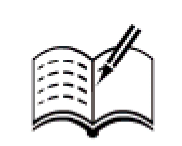
1.3.4 WiFi 后台自动搜索功能

Android 4.3 系统的“WiFi 高级设置”中，增加了“总是自动搜索 WiFi 信号”（Scanning always available）的选项，并且处于默认开启状态，它可令手机在进入 WiFi 的区域后立即连接，并能快速实现室内定位功能，较传统的 GPS 省电而精准，不过这项功能有可能影响电池续航时间，如果对导航定位类应用的依赖不是很大，建议关闭。

1.3.5 图形

Android 4.3 系统支持全新的图形接口 OpenGL ES 3.0，相比 OpenGL ES 2.0，它拥有更多的





缓冲区对象，支持 GLSL ES 3.0 着色语言、32 位整数和浮点数据类型操作，统一了纹理压缩格式 ETC，实现了多重渲染目标和多重采样抗锯齿，这将为 Android 游戏带来更加出色的视觉效果。

1.3.6　音频

新版系统对播放音质进行了提升，世界著名的音频和多媒体技术研究机构 Fraunhofer IIS 研发了最新产品 Fraunhofer Cingo，结合安卓设备对 HE-AAC Multichannel（多声道）提供的固有支持，可以为用户带来影院级的掌上环绕声音效体验。

1.3.7　流媒体加密

Android 4.3 拥有各式各样的全新 DRM 执行方式： MPEG DASH、VP8 编码和无须缓存的表面编码，并可通过 MPEG-4 媒体流合并器来融合视频与音频文件的输出。

1.3.8　通知栏

在 Android 4.3 系统中，所有使用“前台服务接口”实现运行的应用程序都会被强制显示在通知栏和拓展通知窗的“正在运行”（Ongoing）中，而不受“显示通知”选项的控制。这种做法的目的是让无法杀掉进程且在后台静默运行的应用程序在通知栏上“显形”，让用户得以关注“行为不端”的应用程序。

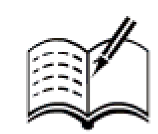
1.3.9　相机

在 JellyBean 系列系统中，“全景拍摄”包含两种模式：“横轴全景”（Panorama）和“360 度全景”（Photosphere）。其中，后者逼真的“街景浏览”效果和“鱼眼浏览”效果着实惊艳了许多用户；不过，拍摄过程中产生的图片衔接与校准问题，又令不少人郁闷。好消息是，谷歌地图项目经理 Evan Rapoport 宣布已大大提升了 Android 4.3 的全景拍照功能，通过优化对准和拼接颜色，照片将更加明亮，过度将更加自然。接下来，借助 HTML 5 和 Java 技术，可以将自己得意的街景照片分享到论坛和社交网站。

1.3.10　拨号面板

自动补全功能：当用户依据数字或字母而点击宫格按键时，面板界面会实时展示与之对应的联系人建议，再次点击即可完成快速输入，但受空间限制，其最多可在一行里显示前 3 个符合条件的联系人。





Note

1.3.11 键盘与输入

谷歌通过优化算法，调整了键盘，带来了更好的触摸识别和文本输入；此外，新版系统还缓解了操纵游戏手柄时的延迟现象。

1.3.12 设置

在 Android 4.3 系统中，用户可以通过“设置”>“应用”里最右侧的“禁用应用”标签（Disabled tab），直接浏览所有被冻结的自带应用，而不必再通过“所有应用”标签（Alltab）苦苦找寻了。

1.3.13 支持国际用户

Android 4.3 中新增的语言支持包括阿姆哈拉语、南非荷兰语、印地语、斯瓦希里语和祖鲁语，对于希伯来语和亚拉姆语等一些从右至左阅读的文字也提供了更好的支持。

1.3.14 新增多国语言支持

在 Android 4.3 系统中，启动器、时钟、快速设置开关、拨号盘、联系人应用、安装向导、下载应用和其他更多的界面元素都为从右至左的文字显示（RTL）提供了更多的支持，用于 RTL 测试的开发者工具也被加入到 Android SDK 当中。

1.3.15 谷歌套件

Android 除了保持以往的系统版本升级外，还逐渐引入模块化升级，即便用户无法及时将设备系统更新到最新版本，仍然可以通过谷歌商店替换部分符合条件的内置应用，体验到更为人性化的操作。

☑ 谷歌音乐播放器（Google PlayMusic）得到改进，融入了卡片式设计灵感，用色更明亮，交互更流畅。

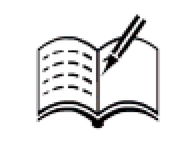
☑ 谷歌云笔记（Google Keep）早在 3 月份就正式推出，而今整合到新版系统中，方便用户随时随地记录感悟。

☑ 环聊（Hangouts）于 2013 年谷歌 I/O 大会上正式推出，脱胎于 Google Plus，整合了多项谷歌聊天工具，用以替代 Google Talk 应用。

1.3.16 其他新增特性

Android 4.3 系统一定程度上提升了电池的续航能力，设备待机时间有所延长；新内核调整





了显示颜色，使得 Nexus 4 的屏幕不再如水洗一般清淡；另外，新基带增强了信号的稳定性，缓解了部分 Nexus 设备的 WiFi 和 2G 数据网络的延迟问题。

1.4 如何学习 Android

1.4.1 如何学好 Android

如何学好 Android，这是所有初学者共同面对的问题，其实每种新技术的学习方法都大同小异，需要注意的主要有以下几点。

☑ 具备一定的 Java 基础和 XML 基础，因为 Android 应用层的开发一般都是使用 Java 语言编写的，而 Android 界面布局通常都是在 XML 文件中完成的。

☑ 明确自己的学习目标和大的方向，按照自己的学习方向努力学习和认真研究。

☑ 初学者不要看太多的书，先找本基础书系统地学习。很多程序开发人员工作了很久也只熟悉部分基础而已，而没有系统地学习 Android 的体系结构。

☑ 不要死记硬背。在学习 Android 时，首先需要掌握基本的 Java 语法，并大概了解一些功能，然后可以借助开发工具（如 Eclipse）的代码辅助功能，完成代码的录入，这样可以快速地进入学习状态。

☑ 多实践，多思考，多请教。光读懂书本中的内容和技术是不行的，必须动手编写程序代码，并运行程序、分析运行结构，从而对学习内容有个整体的认识和肯定。用自己的方式去思考问题，编写代码来提高编程思想。平时多请教老师或经理，与其他人多沟通技术问题，提高自己的技术和见识。

☑ 不要急躁。遇到技术问题，必须冷静对待，不要让自己的大脑思绪混乱，保持清醒的头脑才能分析和解决各种问题。可以尝试听歌、散步等活动放松自己。

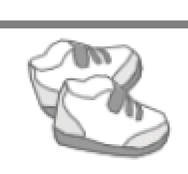
☑ 遇到问题，首先尝试自己解决，这样可以提高自己的程序调试能力，并对常见问题有一定的了解，明白出错的原因，甚至举一反三，解决其他关联的错误问题。

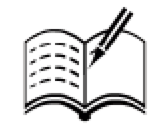
☑ 多查阅资料。可以经常到互联网上搜索相关资料或者解决问题的办法，网络上已经摘录了很多人遇到的问题和不同的解决办法，分析这些解决问题的方法，找出最好、最适合自己的方法。

☑ 多阅读别人的源代码。不但要看懂别人的程序代码，还要分析编程者的编程思想和设计模式，并融为己用。

1.4.2 Android API 文档的使用

API 的全称是 Application Programming Interface，即应用程序编程接口。Android API 文档是 Android 程序开发不可缺少的编程词典，它记录了 Android 编程中海量的 API，主要包括类的继承结构、成员变量和成员方法、构造方法、静态成员的详细说明和描述信息等。可以在 Android







官方网站 http://www.android.com 中找到最新版本的 Android SDK 文档(即 API 文档)。Android API 文档的页面效果如图 1.6 所示。

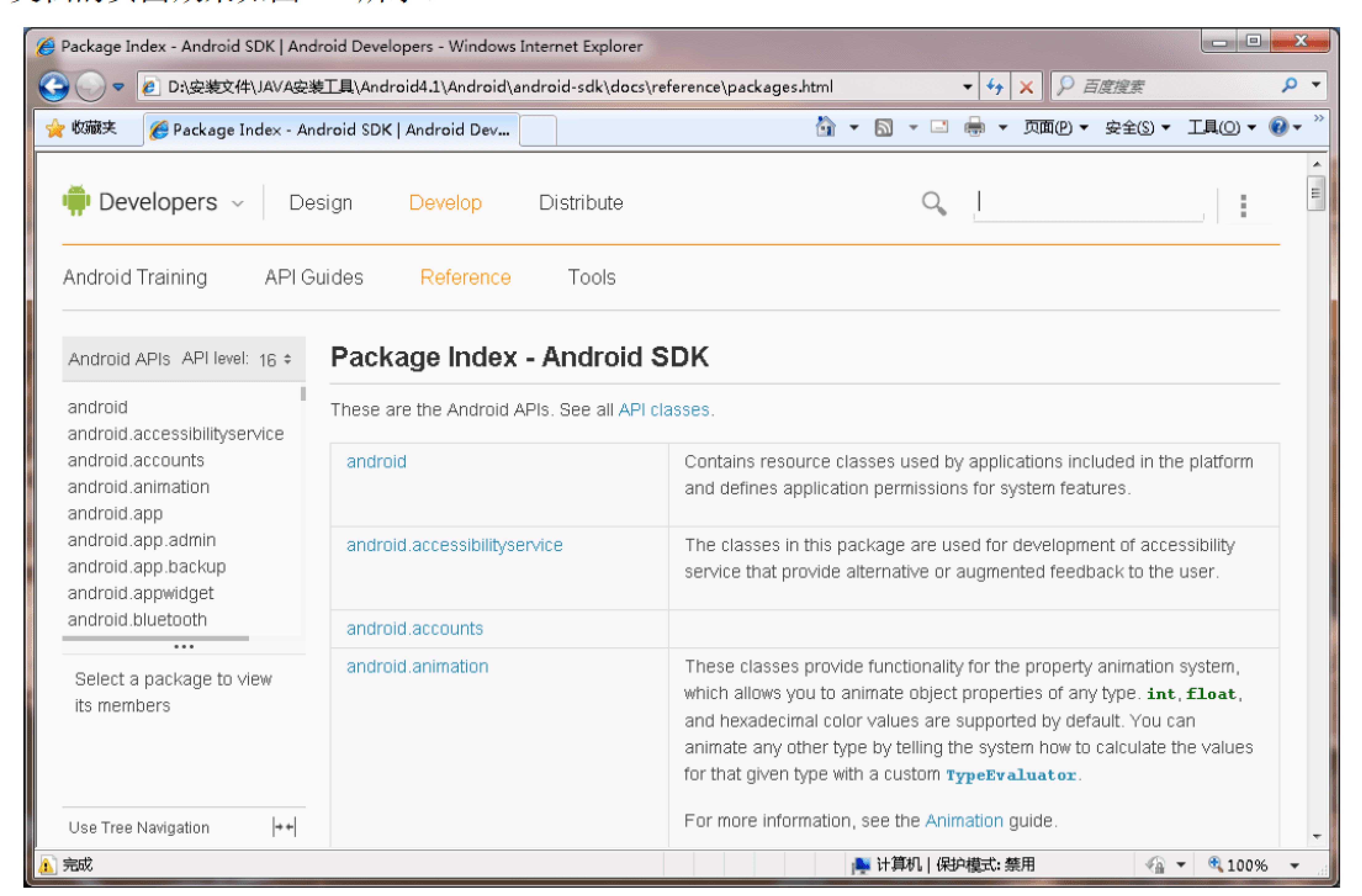


图 1.6　Android API 文档页面效果

说明：

如果已经在本地计算机上下载并安装了 Android 4.3 SDK，则可以通过双击“sdk\docs”文件夹中的 index.html 打开 Android API 文档。

1.5　本章小结

本章简单描述了什么是 Android 并介绍了其相关特性，并重点介绍了最新的 Android 4.3 版本新增的一些特性。通过本章的学习，读者应该能够了解什么是 Android 和它的不同版本以及如何学习 Android，并熟悉 Android 4.3 中新增的特性。

第 2 章

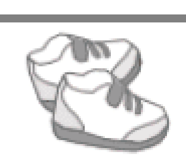
搭建 Android 开发环境

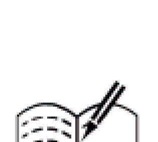
（视频讲解：1 小时 26 分钟）

随着移动设备的不断普及发展，相关软件的开发也越来越受到程序员的青睐。目前移动开发领域，以 Android 发展最为迅猛，本章将通过 Android 开发环境的搭建和第一个 Android 程序的开发过程，带领读者进入 Android 程序开发的世界。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 下载 JDK
- ☐ 安装并配置 JDK
- ☐ 下载 ADT Bundle
- ☐ 创建 Android 应用程序
- ☐ 创建 AVD 模拟器
- ☐ 创建一个可以运行在所有 Android 版本上的程序
- ☐ 在 Android 窗口中输出“你好”中文字符串
- ☐ 将应用程序的名称修改为 Android





2.1　搭建 Android 开发环境

学习一种新技术之前，初学者经常会听到老师强调“工欲善其事，必先利其器”这句话。在学习 Android 开发之前，必须首先熟悉并搭建它所需要的开发环境，本节将对如何搭建 Android 开发环境进行详细讲解。

2.1.1　Android 开发准备

本节讲述使用 Android SDK 进行开发所必需的硬件和软件需求。对于硬件方面，要求 CPU 和内存尽量大。Android 4.3 SDK 下载大概需要 700MB 硬盘空间。由于开发过程中需要反复重启模拟器，而每次重启都会消耗几分钟的时间（视机器配置而定），因此使用高配置的机器能节约不少时间。

下面重点讲解一下软件需求，这里将介绍两个方面：操作系统和开发环境。

☑　操作系统要求

支持 Android SDK 的操作系统及其要求如表 2.1 所示。

表 2.1　Android SDK 对操作系统的要求

操 作 系 统	要　　求
Windows	Windows XP（32 位） Windows 7（32 位或 64 位） Windows 8（32 位或 64 位）
Mac OS	10.5.8 或更新（仅支持 x86）
Linux（在 Ubuntu 的 10.04 版测试）	需要 GNU C Library (glibc) 2.7 或更新 在 Ubuntu 系统上，需要 8.04 版或更新 64 位版本必须支持 32 位应用程序

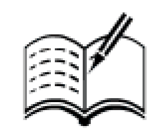
☑　开发环境要求

在安装 Android 应用程序之前，首先搭建好 Android 开发所需要的开发工具，本书以 Windows 7 操作系统为例讲解 Android 的开发。Android 开发所需的软件及其下载地址如表 2.2 所示。

表 2.2　Android 开发所需的软件及下载地址

软 件 名 称	下 载 地 址	本书使用的版本
JDK	http://www.oracle.com	JDK 7 Update 10
ADT Bundle	http://www.android.com	adt-bundle-windows-x86-20130917







说明：

ADT Bundle 是 Google 公司为 Android 开发人员提供的一个集成开发工具，包括了 Eclipse、Android SDK 以及 ADT 开发组件，而 ADT 开发组件已经自动集成到了 Eclipse 开发环境中，无须用户手动安装。

2.1.2　JDK 的下载

由于 Sun 公司已经被 Oracle 收购，因此 JDK 可以在 Oracle 公司的官方网站（http://www.oracle.com/cn/index.html）下载。下面以 JDK 7 Update 10 版本为例介绍下载 JDK 的方法，其具体步骤如下：

（1）打开浏览器，进入 Oracle 官方主页，地址是 http://www.oracle.com/index.html，如图 2.1 所示。

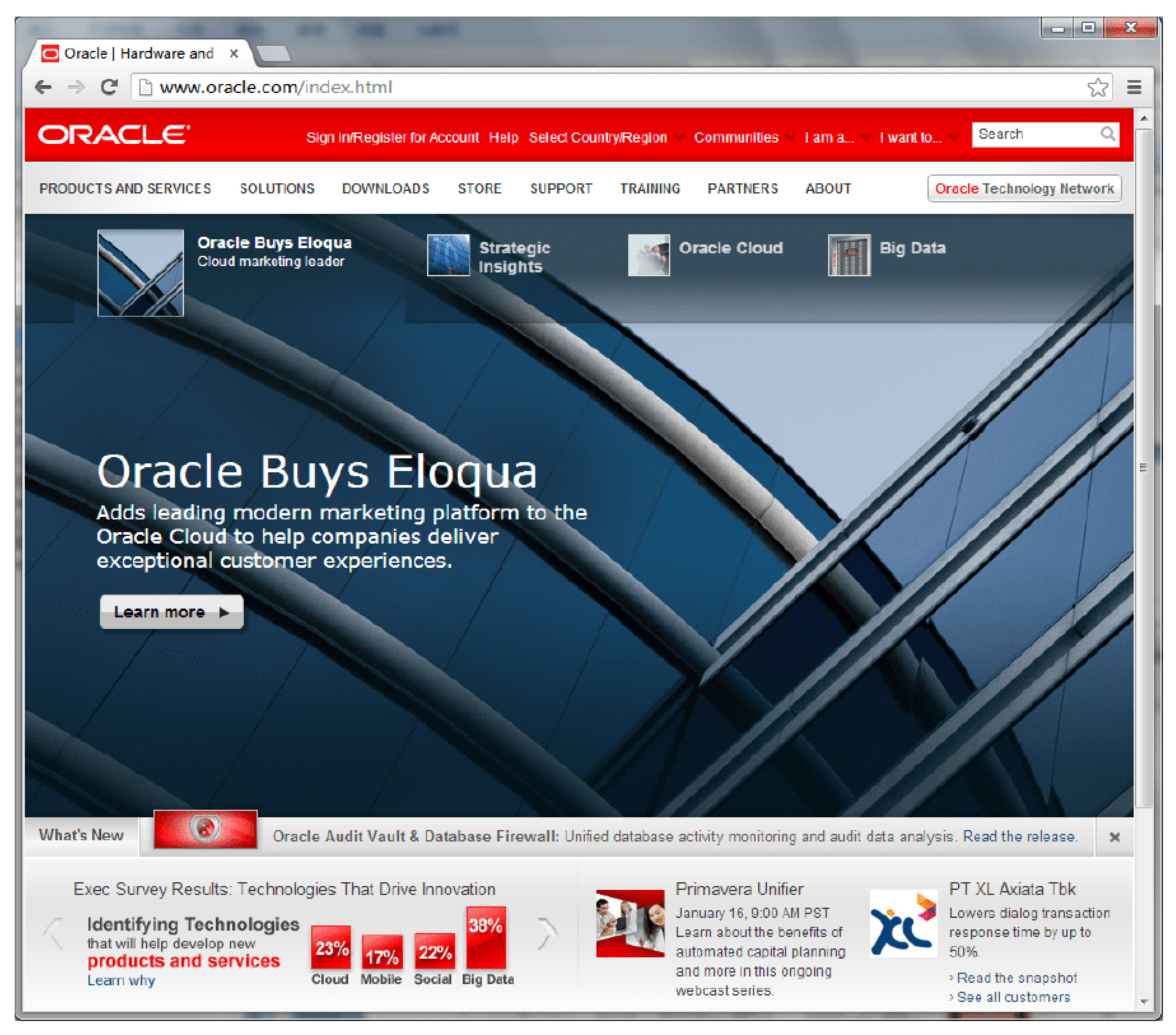
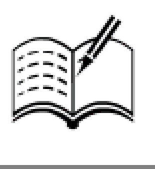


图 2.1　Oracle 官方主页

（2）选择 DOWNLOADS/Java for Developers 菜单，在跳转的页面中滚动到如图 2.2 所示的位置。

（3）单击 JDK 下方的 DOWNLOAD 按钮，将进入如图 2.3 所示的页面。

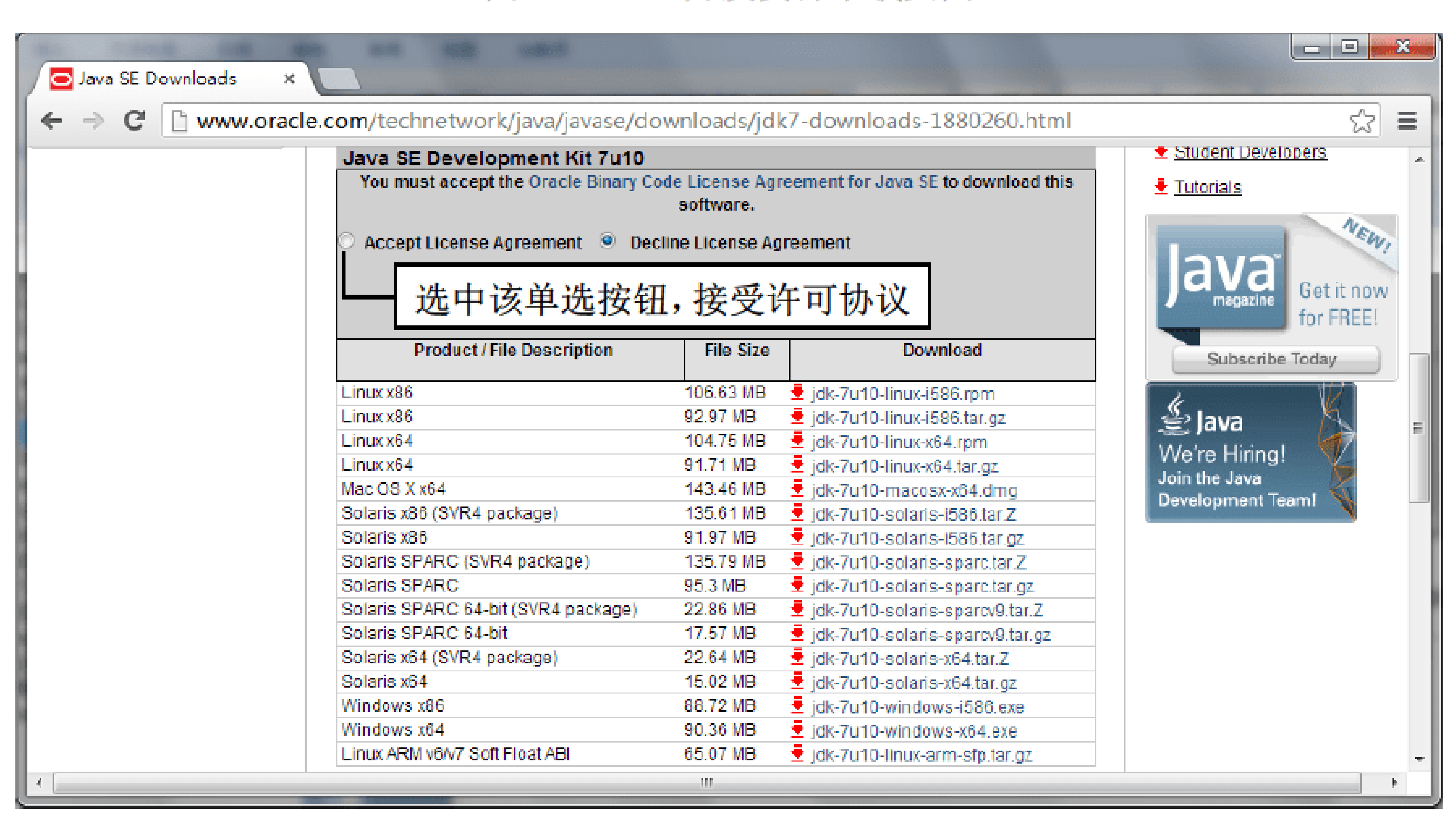




Note



图 2.2　Java 开发资源下载页面



Product / File Description	File Size	Download
Linux x86	106.63 MB	jdk-7u10-linux-i586.rpm
Linux x86	92.97 MB	jdk-7u10-linux-i586.tar.gz
Linux x64	104.75 MB	jdk-7u10-linux-x64.rpm
Linux x64	91.71 MB	jdk-7u10-linux-x64.tar.gz
Mac OS X x64	143.46 MB	jdk-7u10-macosx-x64.dmg
Solaris x86 (SVR4 package)	135.61 MB	jdk-7u10-solaris-i586.tar.Z
Solaris x86	91.97 MB	jdk-7u10-solaris-i586.tar.gz
Solaris SPARC (SVR4 package)	135.79 MB	jdk-7u10-solaris-sparc.tar.Z
Solaris SPARC	95.3 MB	jdk-7u10-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	22.86 MB	jdk-7u10-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	17.57 MB	jdk-7u10-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	22.64 MB	jdk-7u10-solaris-x64.tar.Z
Solaris x64	15.02 MB	jdk-7u10-solaris-x64.tar.gz
Windows x86	88.72 MB	jdk-7u10-windows-i586.exe
Windows x64	90.36 MB	jdk-7u10-windows-x64.exe
Linux ARM v6/v7 Soft Float ABI	65.07 MB	jdk-7u10-linux-arm-sfp.tar.gz

图 2.3　JDK 下载页面

（4）选中 Accept License Agreement 单选按钮，接受许可协议，并根据计算机硬件和系统而选择适当的版本进行下载，如图 2.4 所示。



说明：

如果用户的系统是 Windows 32 位，那么下载 jdk-7u10-windows-i586.exe；如果是 Windows 64 位的系统，那么下载 jdk-7u10-windows-x64.exe。



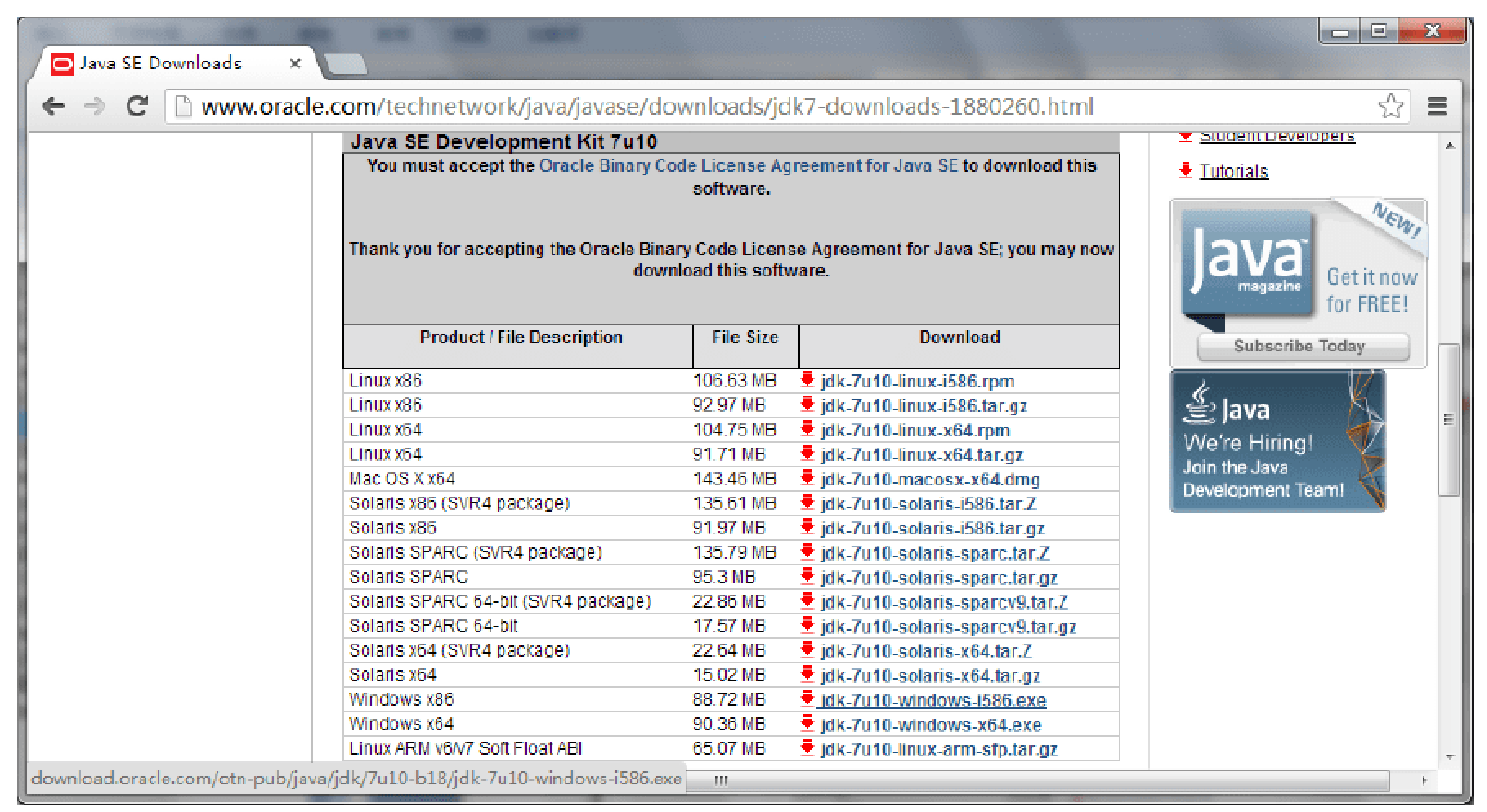


图 2.4　接受许可协议并下载

2.1.3　JDK 的安装与配置

JDK 的安装文件下载后，就可以安装 JDK 了，其具体的安装步骤如下：

（1）双击刚刚下载的安装文件，将弹出如图 2.5 所示的欢迎对话框。

（2）单击“下一步”按钮，将弹出“自定义安装”对话框，在该对话框中，可以选择安装的功能组件，这里选择默认设置，如图 2.6 所示。



图 2.5　欢迎对话框

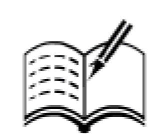
图 2.6　“自定义安装”对话框

（3）单击“更改”按钮，将弹出“更改文件夹”对话框，在该对话框中将 JDK 的安装路径更改为 K:\Java\jdk1.7.0_10\，如图 2.7 所示，单击“确定”按钮，将返回到“自定义安装”对话框中。

（4）单击“下一步”按钮，开始安装 JDK。在安装过程中会弹出 JRE 的“目标文件夹”对话框，这里更改 JRE 的安装路径为 K:\Java\jre7\，然后单击“下一步”按钮，安装向导会继续完

Note





Note

成安装进程。

说明：

JRE 全称为 Java Runtime Environment，它是 Java 运行环境，主要负责 Java 程序的运行，而 JDK 包含了 Java 程序开发所需要的编译、调试等工具，另外还包含了 JDK 的源代码。

（5）安装完成后，将弹出如图 2.8 所示的对话框，单击“关闭”按钮即可。



图 2.7　更改 JDK 的安装路径对话框



图 2.8　“完成”对话框

安装完 JDK 以后，还需要在系统的环境变量中进行配置，其具体方法如下：

（1）在“开始”菜单的“计算机”图标上单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，在弹出的“属性”对话框左侧单击“高级系统设置”超链接，将出现如图 2.9 所示的“系统属性”对话框。

（2）单击“环境变量”按钮，将弹出“环境变量”对话框，如图 2.10 所示。单击“系统变量”栏中的“新建”按钮，创建新的系统变量。

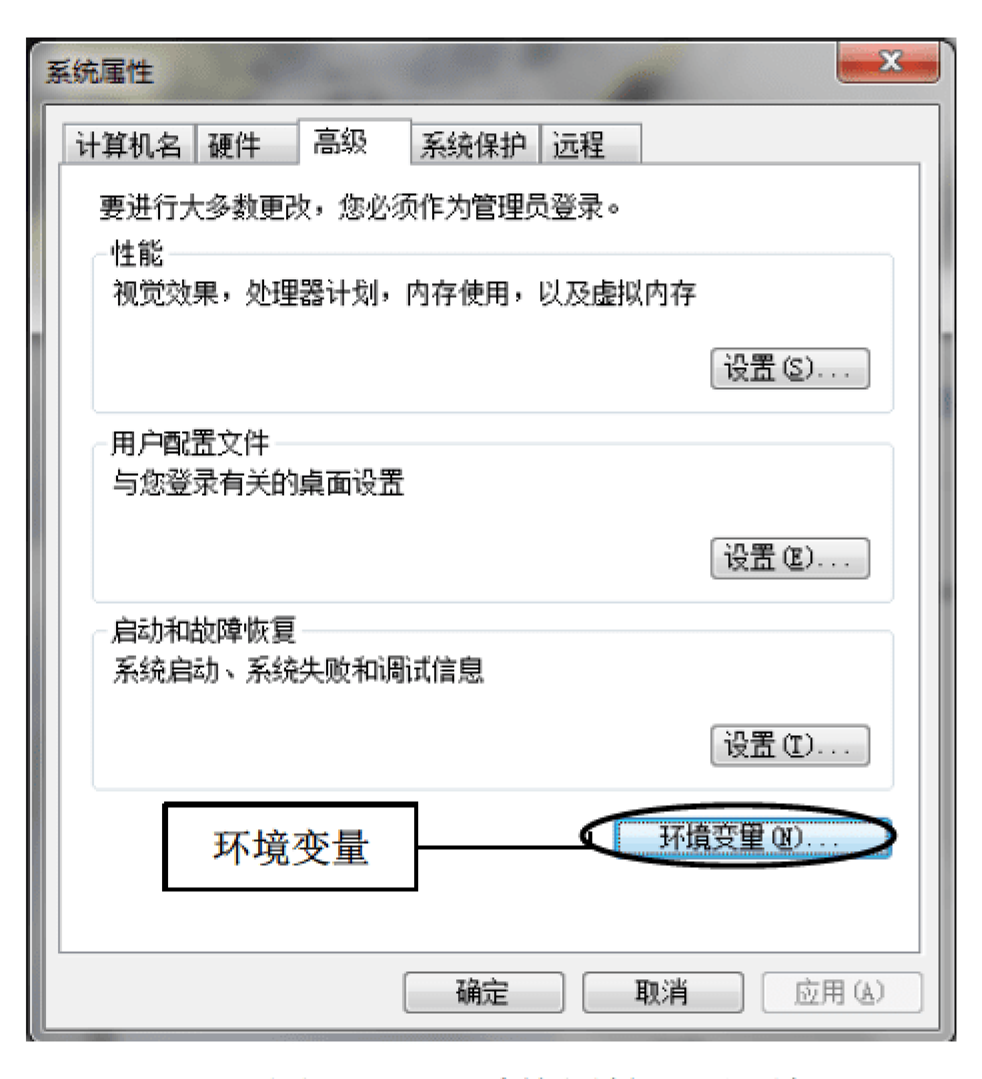


图 2.9　“系统属性”对话框

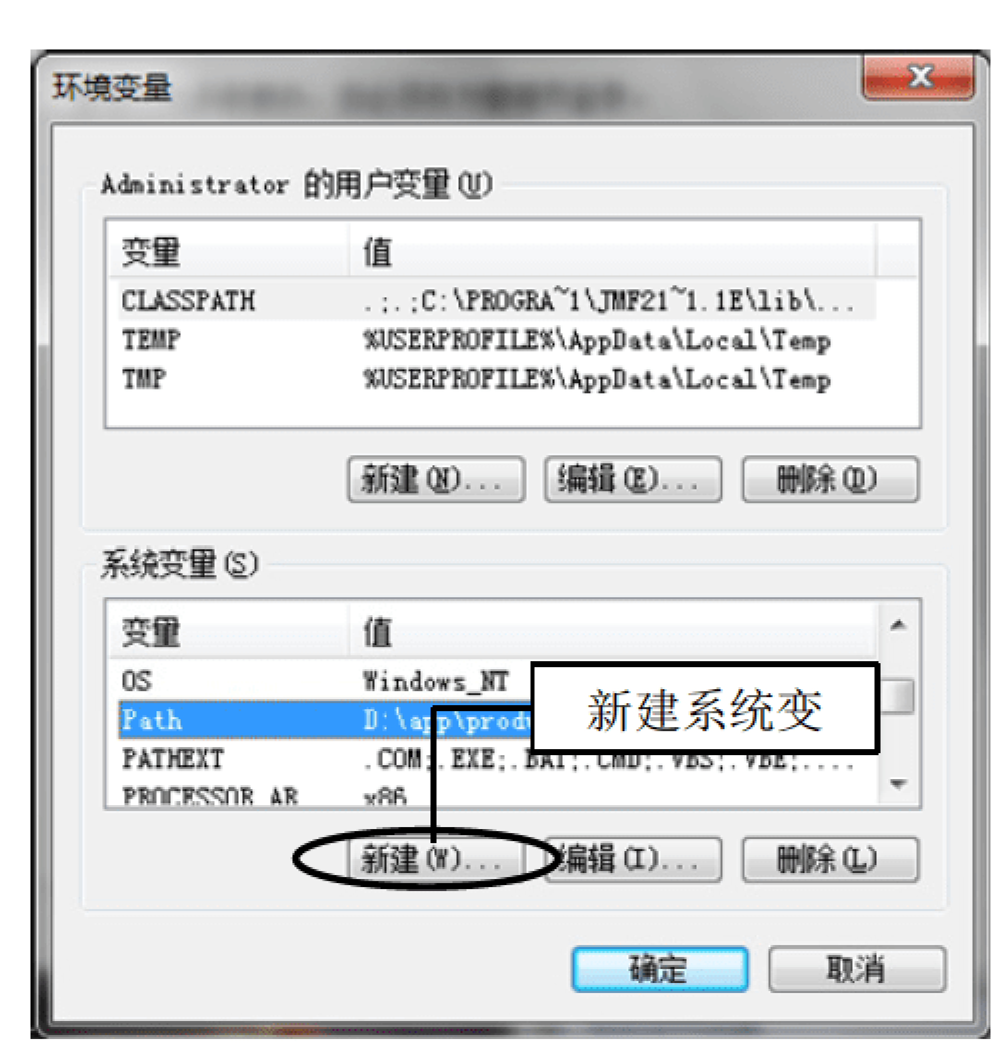
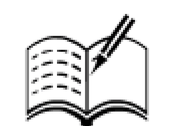


图 2.10　“环境变量”对话框





（3）弹出“新建系统变量”对话框，分别输入变量名“JAVA_HOME”和变量值，其中变量值是笔者的 JDK 安装路径，读者需要根据自己的计算机环境进行修改，如图 2.11 所示。单击“确定”按钮，关闭“新建系统变量”对话框。

（4）在图 2.10 所示的“环境变量”对话框中双击 Path 变量对其进行修改，在原变量值最前端添加“.;%JAVA_HOME%\bin;”变量值（注意：最后的“;”不要丢掉，它用于分割不同的变量值），如图 2.12 所示。单击“确定”按钮完成环境变量的设置。





图 2.11　“新建系统变量”对话框



图 2.12　设置 Path 环境变量值

注意：

不能删除系统变量 Path 中的原有变量值，并且“%JAVA_HOME%\bin”与原有变量值之间用英文半角的“;”号分隔，否则会产生错误。

（5）JDK 安装成功之后必须确认环境配置是否正确。在 Windows 系统中测试 JDK 环境需要选择“开始”/“运行”命令（没有“运行”命令可以按 Windows+R 快捷键），然后在“运行”对话框中输入“cmd”并单击“确定”按钮启动控制台。在控制台中输入“javac”命令，按 Enter 键，将输出如图 2.13 所示的 JDK 的编译器信息，其中包括修改命令的语法和参数选项等信息。这说明 JDK 环境搭建成功。

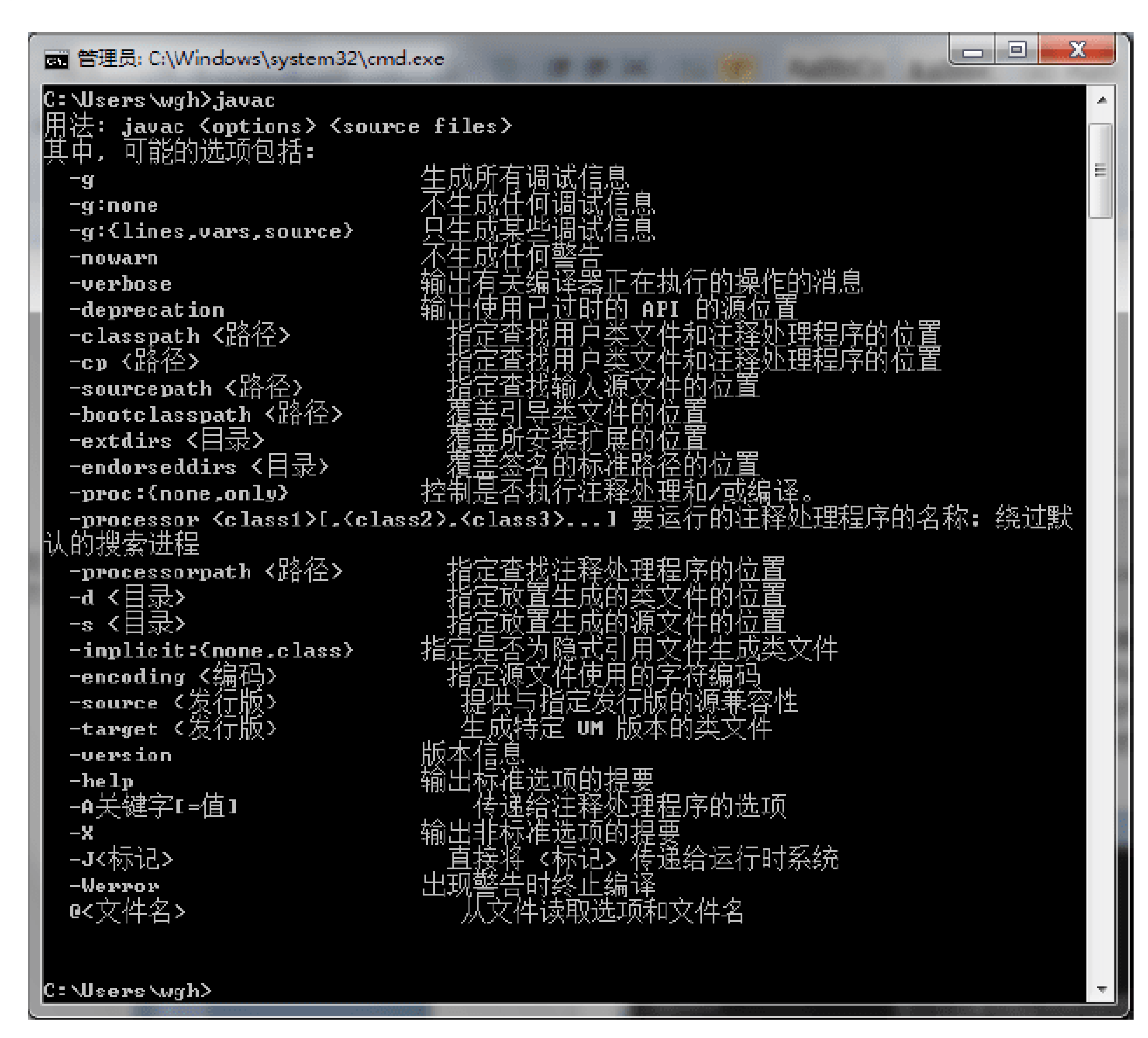
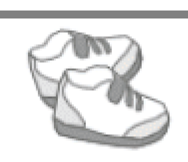


图 2.13　JDK 的编译器信息





Note

2.1.4 ADT Bundle 的下载

Android 程序的开发需要使用 Eclipse 开发工具、Android SDK 和 ADT 组件，Google 公司为了方便开发者使用，将这 3 种工具进行了集成打包，即 ADT Bundle，开发人员只要在 Android 官方网站下载 ADT Bundle 并解压之后，即可使用其中提供的 Eclipse 工具开发 Android 应用。下面介绍 ADT Bundle 的下载过程。

下载 ADT Bundle 的步骤如下：

（1）打开 IE 浏览器，输入网址“http://www.android.com”，浏览 Android 主页，在该主页中，单击 Developers 超链接，如图 2.14 所示。

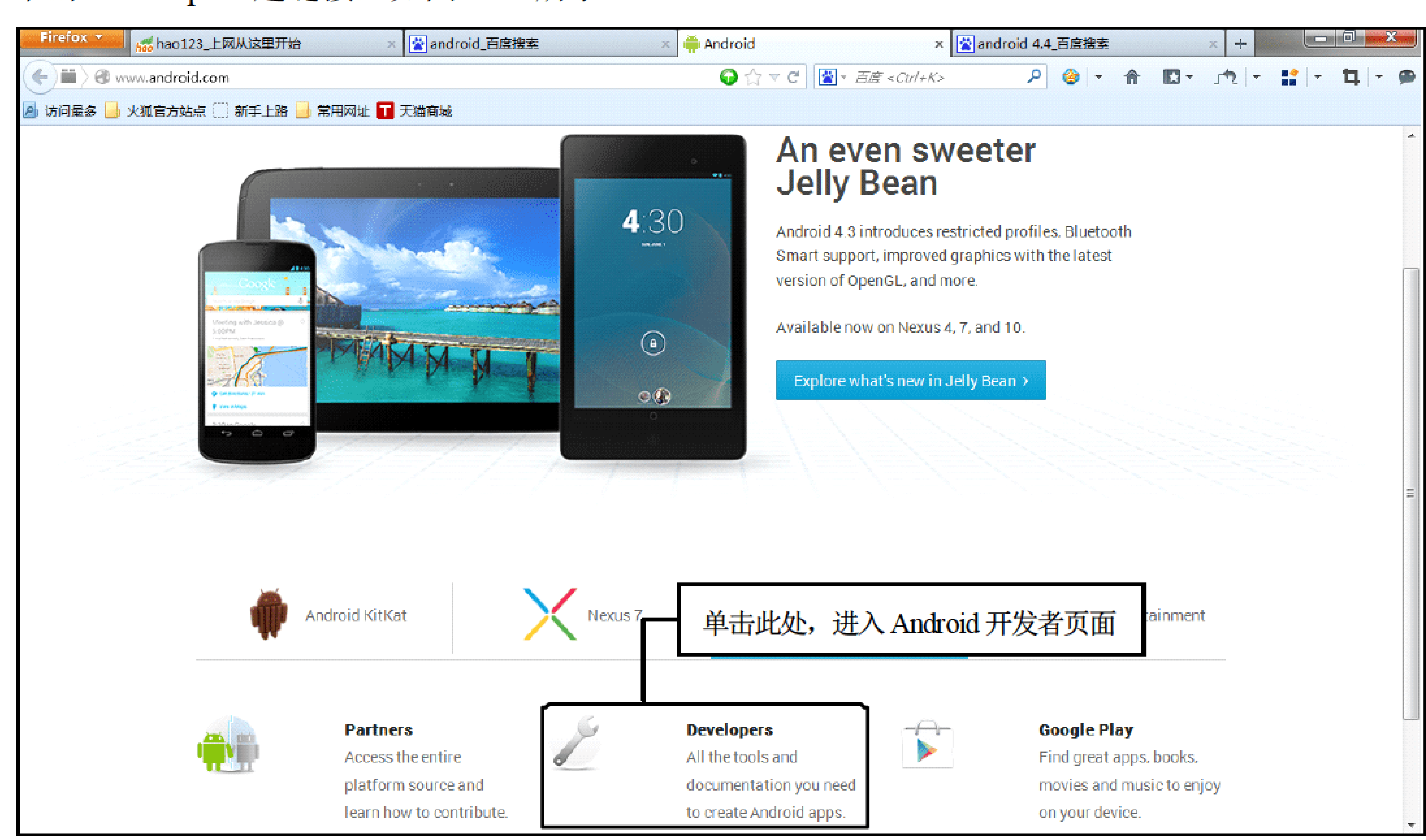


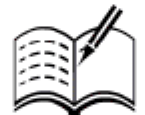
图 2.14　Android 主页

（2）打开 Android Developers 页面，在该页面中以幻灯片形式显示出 Android 4.3 操作系统的相关信息及应用，如图 2.15 所示，单击网页下方的 Get the SDK 超链接。

（3）进入 Android SDK 下载页面，该页面中默认提供 Windows 平台下的 Android SDK 下载链接，如图 2.16 所示。

（4）单击 Download the SDK 按钮，进入 Get the Android SDK 页面，如图 2.17 所示。该页面中显示用户许可协议，选中 I have read and agree with the above terms and conditions 复选框，并选中 32-bit 或者 64-bit 单选按钮，单击 Download the SDK ADT Bundle for Windows 按钮，即可下载指定平台下的 ADT Bundle 组件。





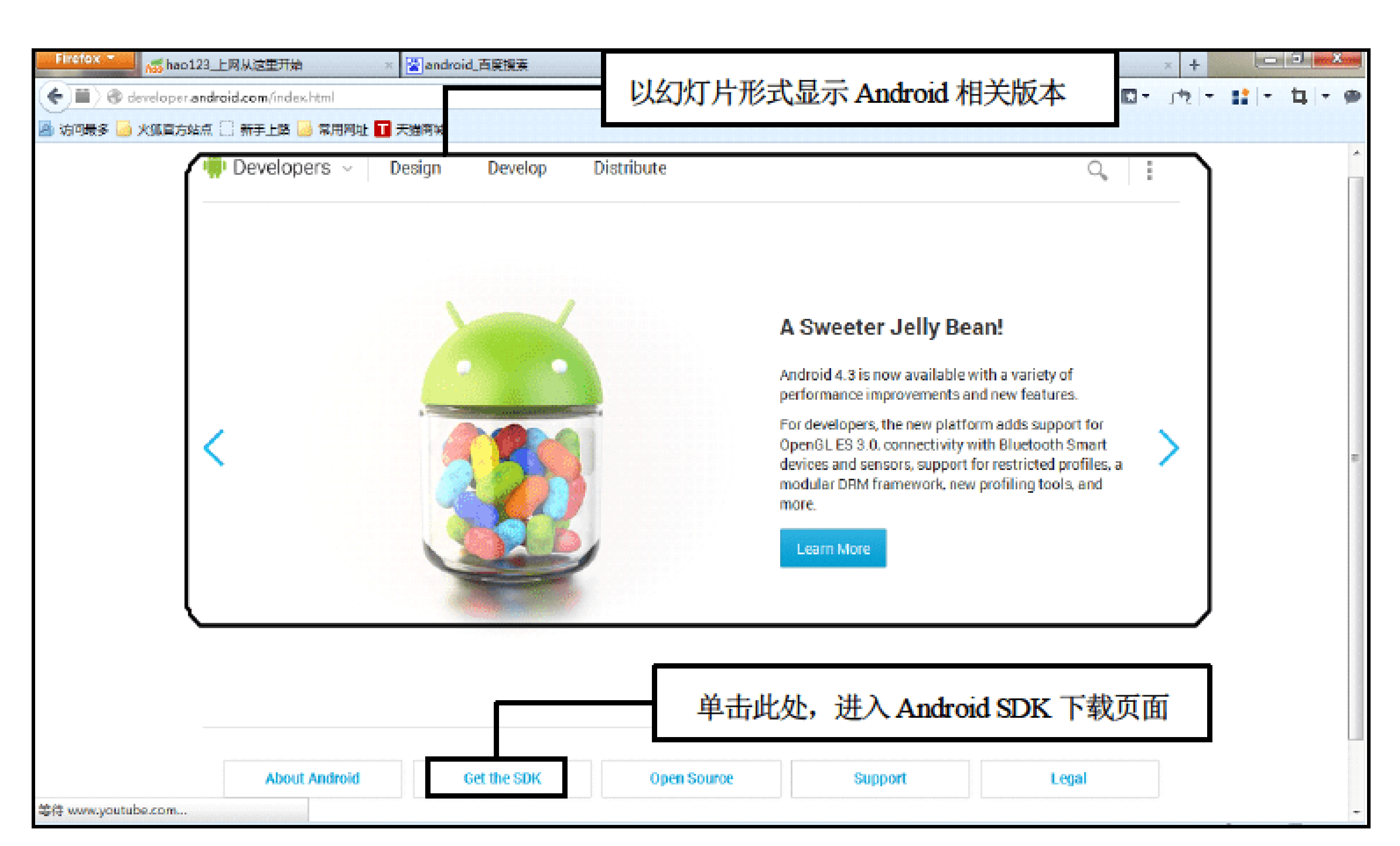


图 2.15　Android Developers 页面



图 2.16　默认的 Android SDK 下载页面

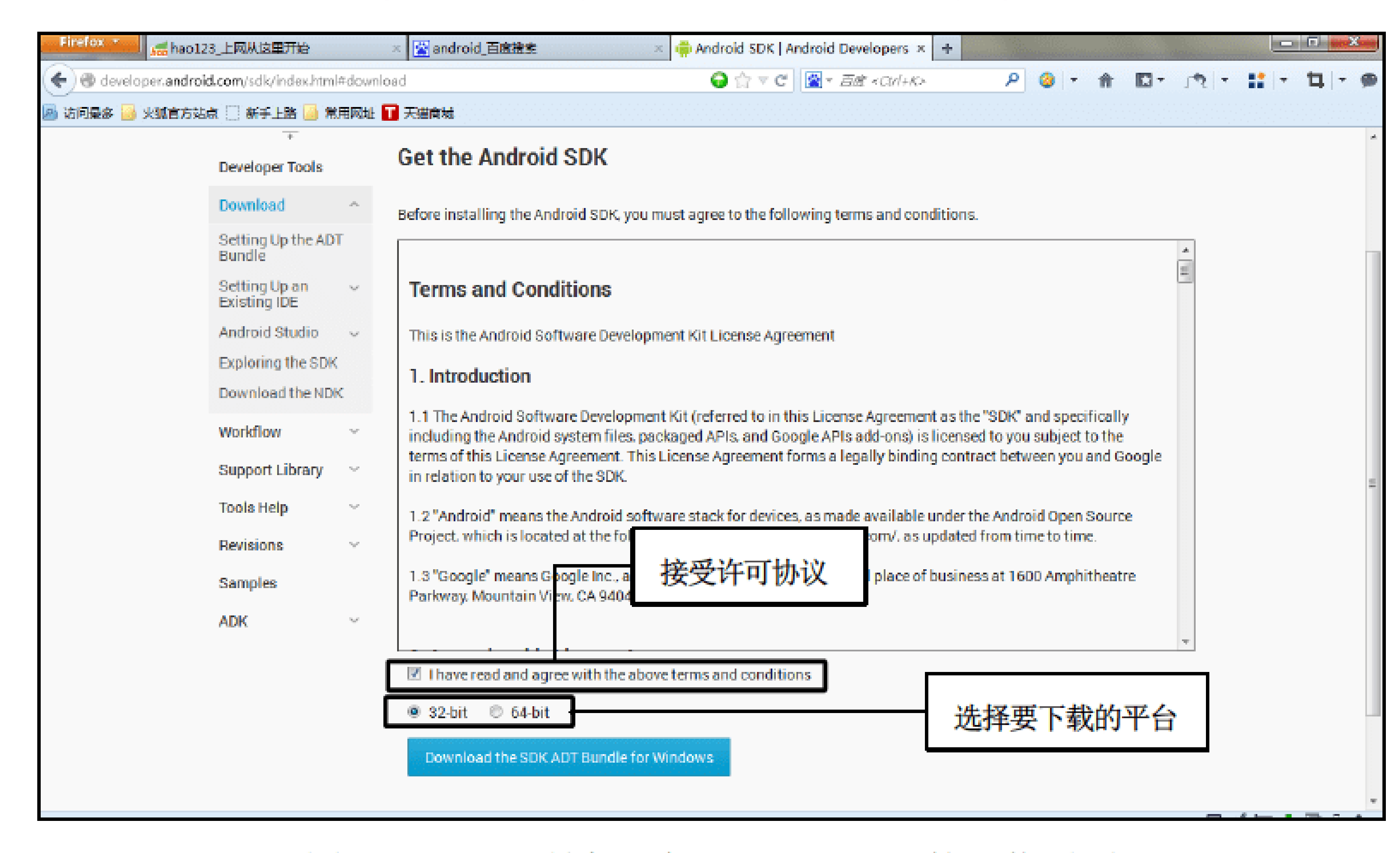


图 2.17　显示所有平台 Android SDK 的下载页面



Note

下载 Windows 32 位平台下的 Android SDK 安装文件 adt-bundle-windows-x86-20130917.zip 后，将该压缩文件解压，解压后的文件夹中包括 eclipse 和 sdk 两个文件夹，以及一个 ADK Manager.exe 文件，其中，eclipse 文件夹中存放的是 Eclipse 开发工具，sdk 文件夹中存放的是 Android 4.3 的开发工具包。解压后的文件夹效果如图 2.18 所示。

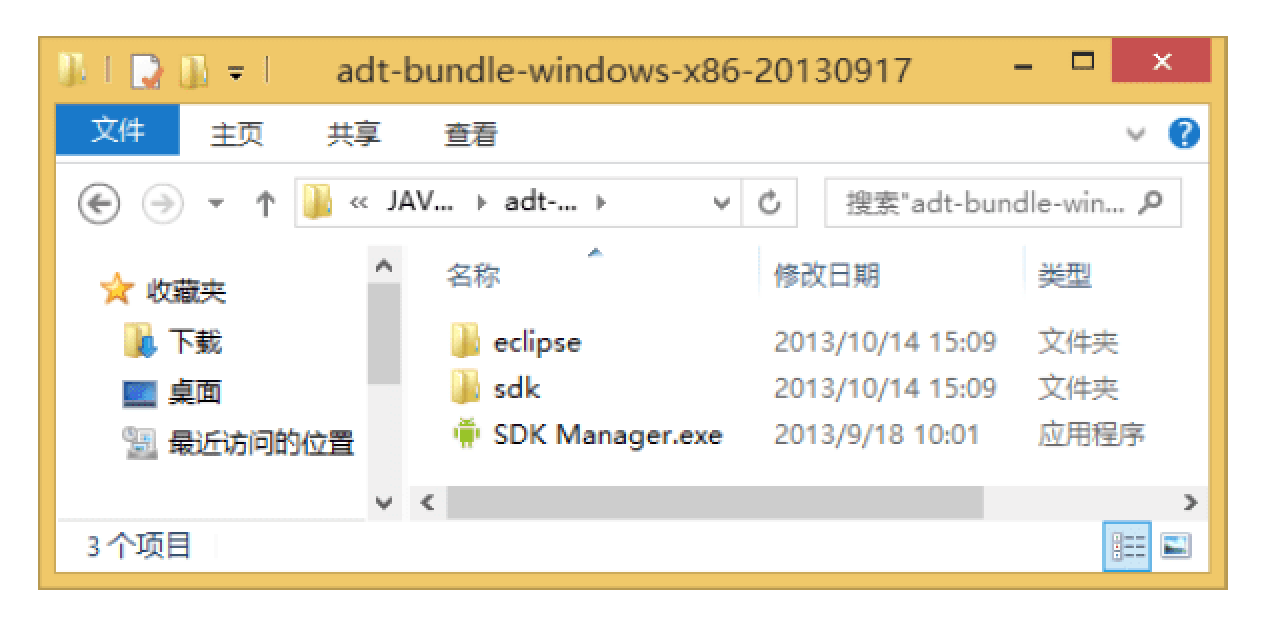


图 2.18　adt-bundle-windows-x86-20130917.zip 压缩文件解压后的效果

eclipse 文件夹中的资源如图 2.19 所示。

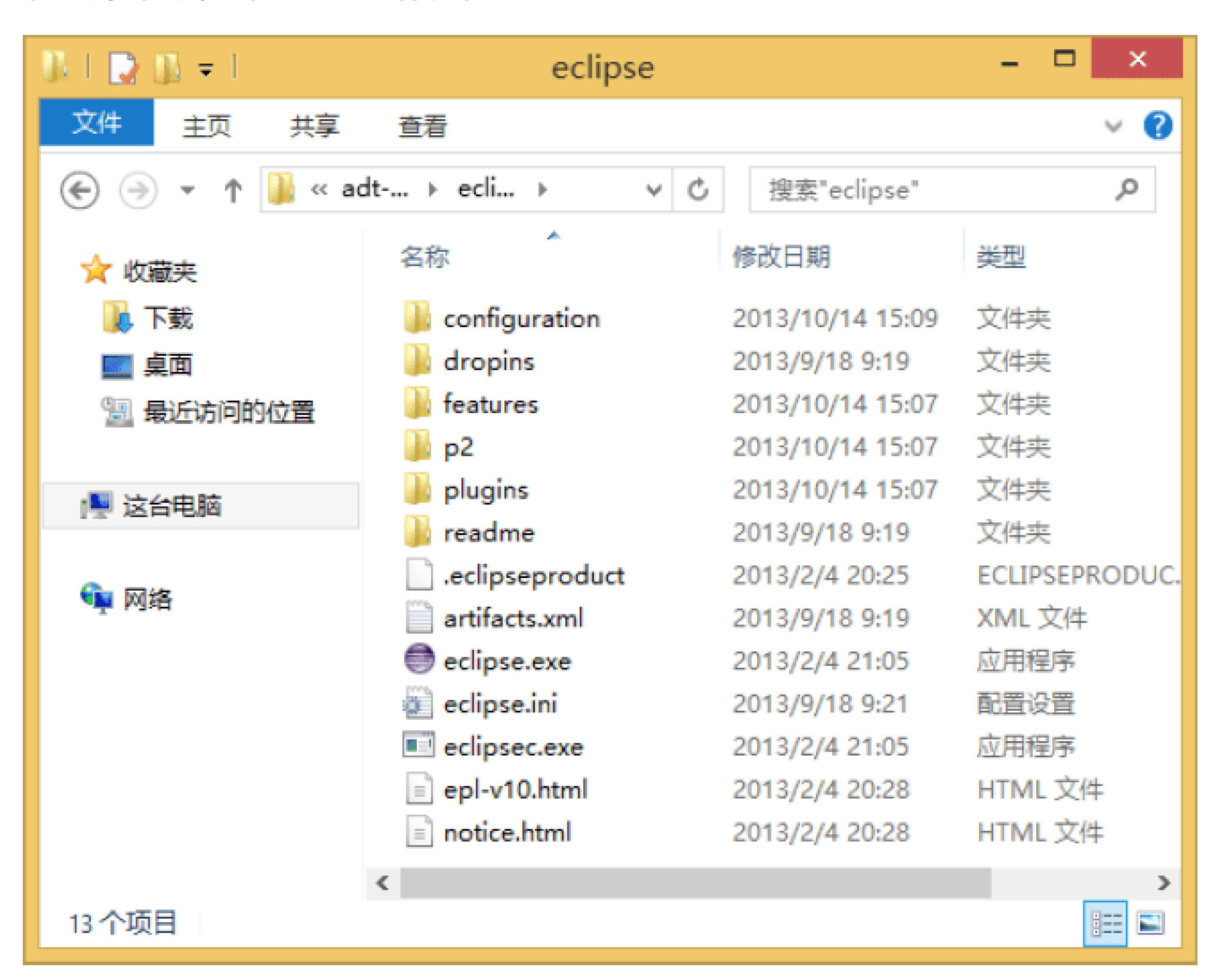


图 2.19　eclipse 文件夹中的资源

在图 2.19 所示的 eclipse 文件中双击 eclipse.exe 文件，即可打开 Android 的开发工具，如图 2.20 所示。

sdk 文件夹中的资源如图 2.21 所示。

通过图 2.21 可以看出，Android SDK 的目录中存在 7 个文件夹，这 7 个文件夹表示的意义分别如下。

☑　add-ons：Android 开发需要的第三方文件。

☑　build-tools：编译选项和相关工具。

☑　extras：附件文档。

☑　platforms：一系列 Android 平台版本。



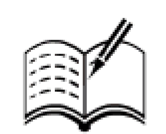




图 2.20　Android 开发工具

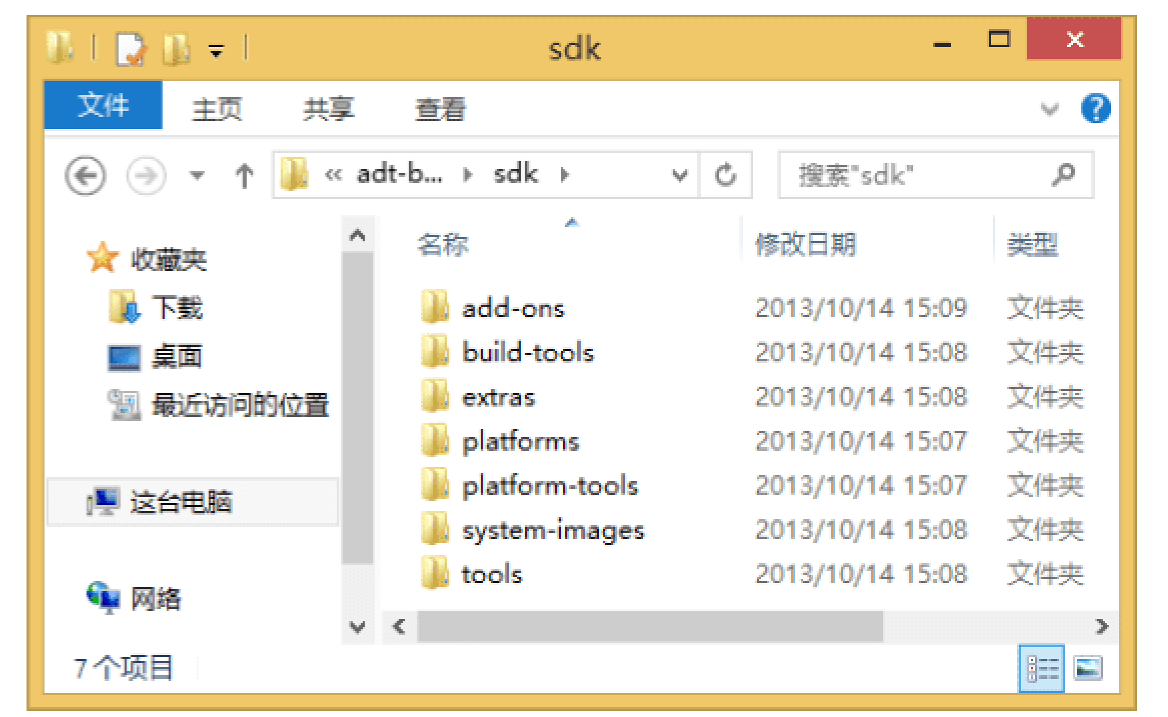


图 2.21　sdk 文件夹中的资源

☑　platform-tools：开发工具，在平台更新时可能会更新。
☑　system-images：系统镜像。
☑　tools：独立于 Android 平台的开发工具，这里的程序可能随时更新。

2.2　第一个 Android 程序

现在开发 Android 程序的环境已经搭建好了，本节将介绍一个简单的 Android 程序的开发过程，让读者对 Android 程序开发流程有一个基本的认识。

2.2.1　创建 Android 应用程序

下面介绍使用 Eclipse 编写本书的第一个 Android 程序的详细步骤。

【例 2.1】 创建 Android 程序的步骤如下：

实例位置：光盘\MR\Instance\02\2.1

（1）双击 eclipse.exe 文件，启动 Android 开发工具，启动后的首页如图 2.22 所示。

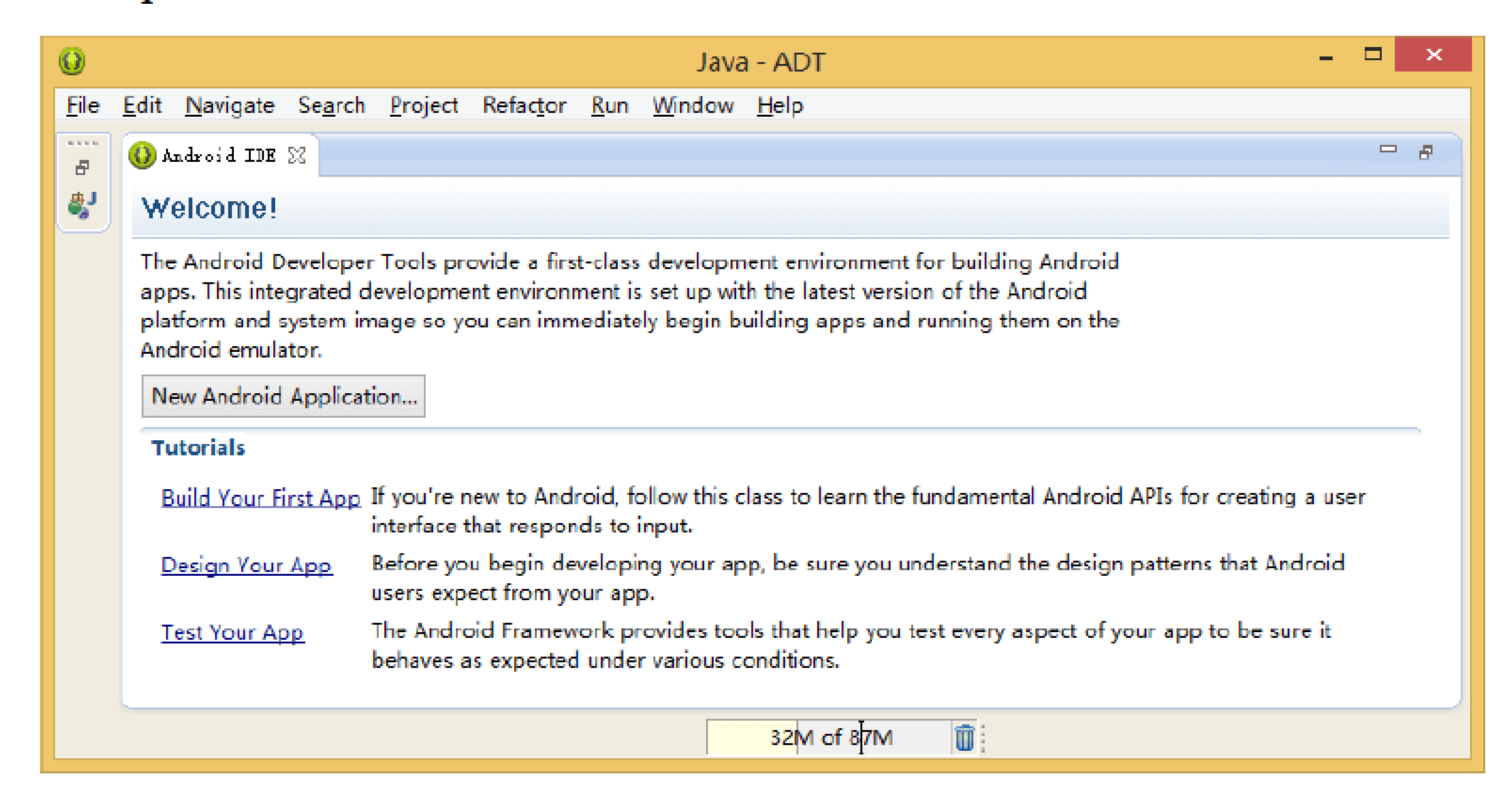


图 2.22　Android 开发工具首页



（2）单击首页的 New Android Application 按钮，或者在菜单栏中依次选择 File/New/Android Application Project 命令，如图 2.23 所示。

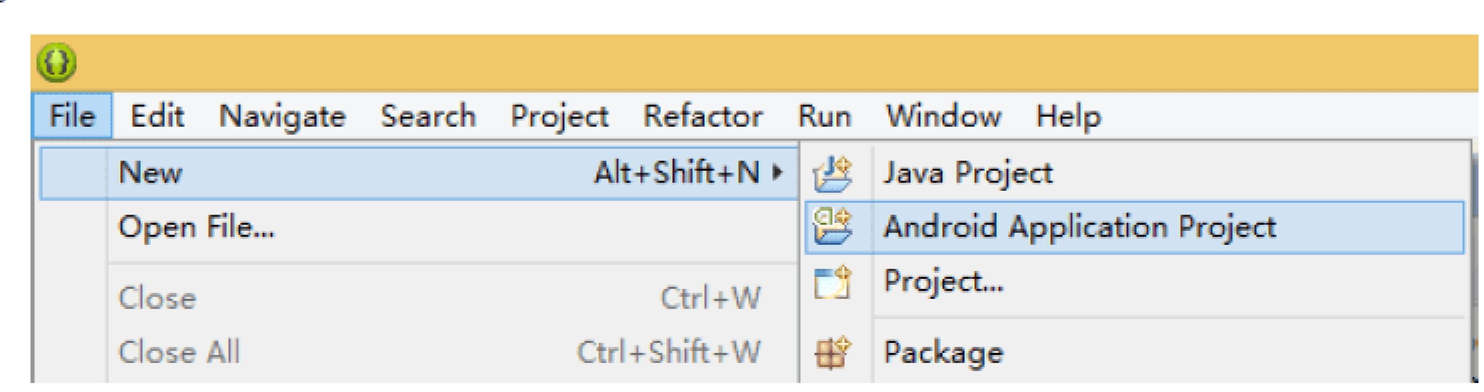


图 2.23　选择 File/New/Android Application Project 命令

（3）弹出 New Android Application 窗口，在该窗口中首先输入项目名称和包名，然后分别在 Minimum Required SDK、Target SDK、Compile With 和 Theme 下拉列表框中选择相应的 Android 版本和主题，如图 2.24 所示。

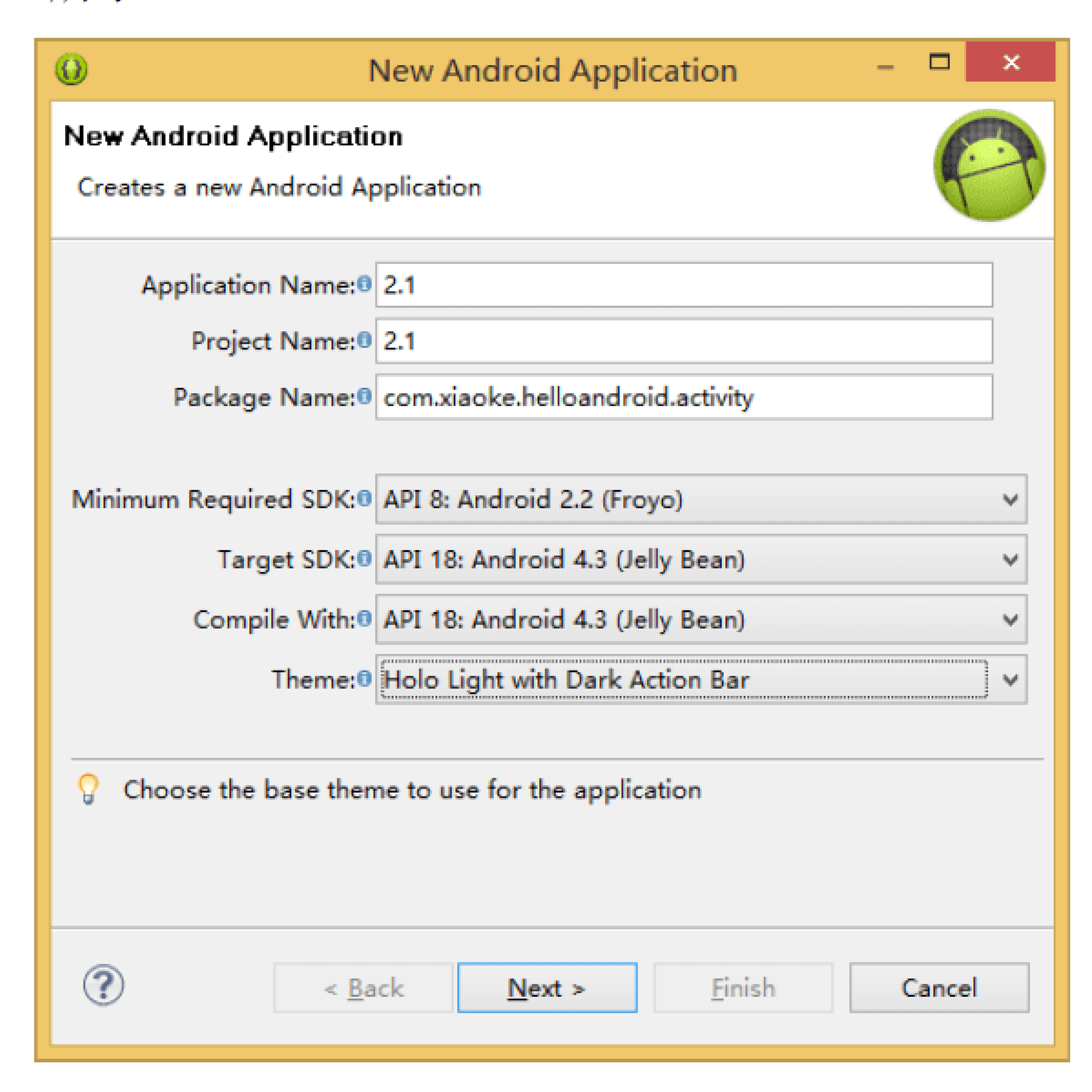


图 2.24　New Android Application 窗口

Note

说明：

在图 2.24 所示的 New Android Application 窗口中有 4 个下拉列表框，分别是 Minimum Required SDK、Target SDK、Compile With 和 Theme，其中，Minimum Required SDK 下拉列表框用来选择 Android 程序可以运行的最低版本，建议选择低版本，这样可以保证创建的 Android 程序能够向下兼容运行；Target SDK 下拉列表框用来选择创建 Android 程序的 Android 版本，建议选择高版本；Compile With 下拉列表框用来选择编译程序所使用的 Android 版本；Theme 下拉列表框用来选择 Android 程序的主题。

（4）在图 2.24 中单击 Next 按钮，进入 Configure Project 界面，该界面中设置是否创建程序图标和 Activity，如图 2.25 所示。



Note

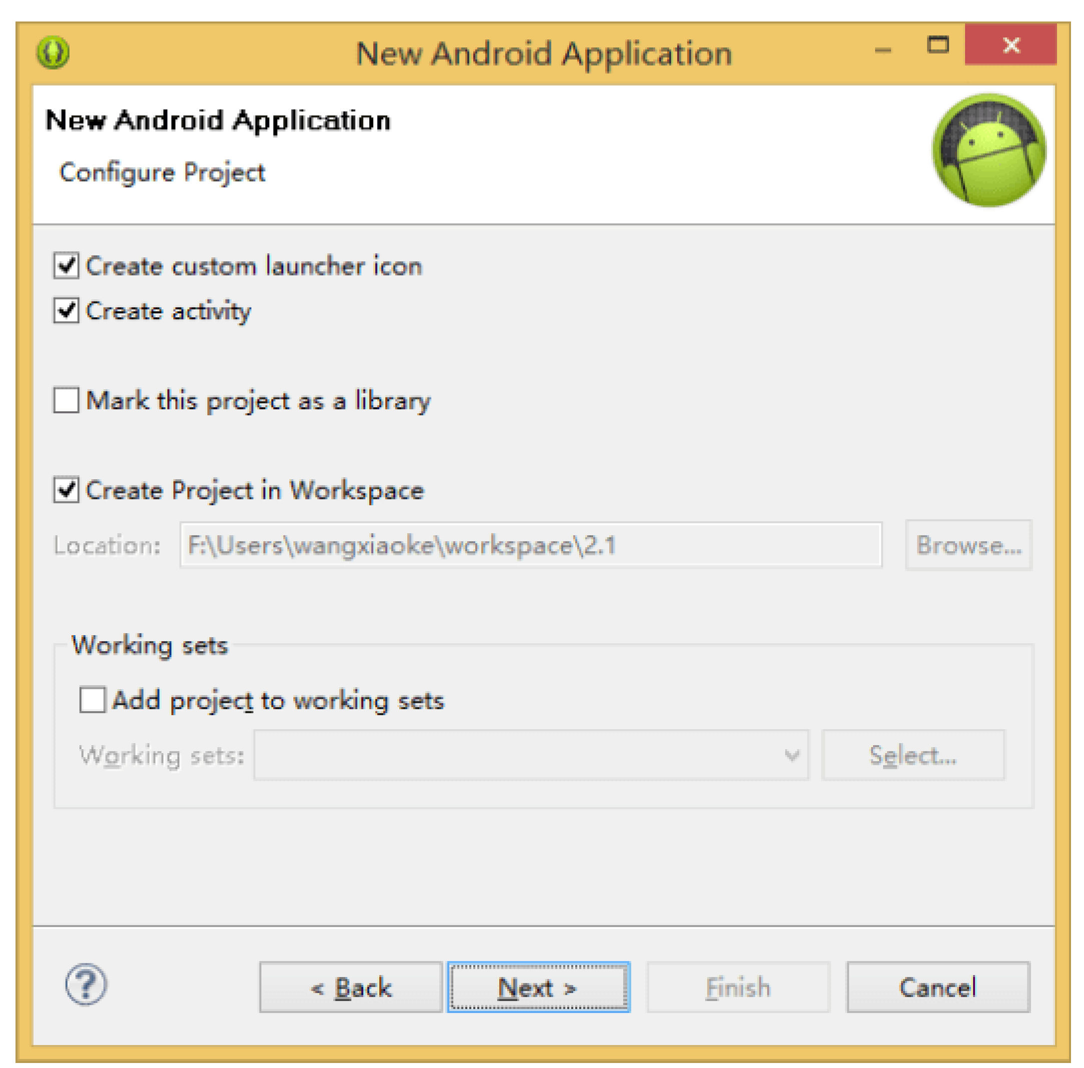


图 2.25 Configure Project 界面

（5）单击 Next 按钮，进入 Configure Launcher Icon 界面，该界面可以对 Android 程序的图标相关信息进行设置，如图 2.26 所示。

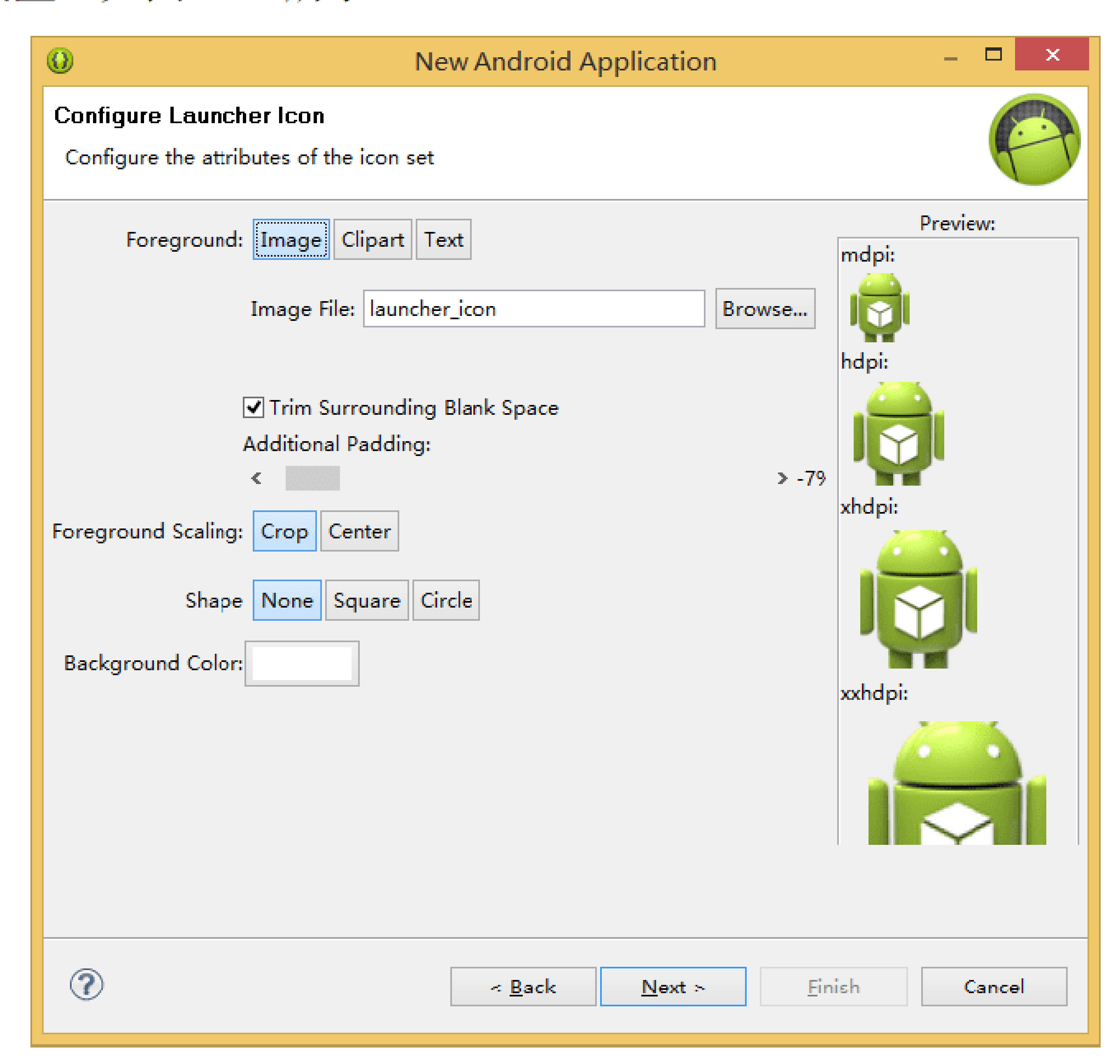


图 2.26 Configure Launcher Icon 界面

（6）单击 Next 按钮，进入 Create Activity 界面，该界面设置要生成的 Activity 的模板，如图 2.27 所示。

（7）单击 Next 按钮，进入 Blank Activity 界面，该界面设置 Activity 的相关信息，包括 Activity 的名称、布局文件名称和导航类型等，如图 2.28 所示。





Note

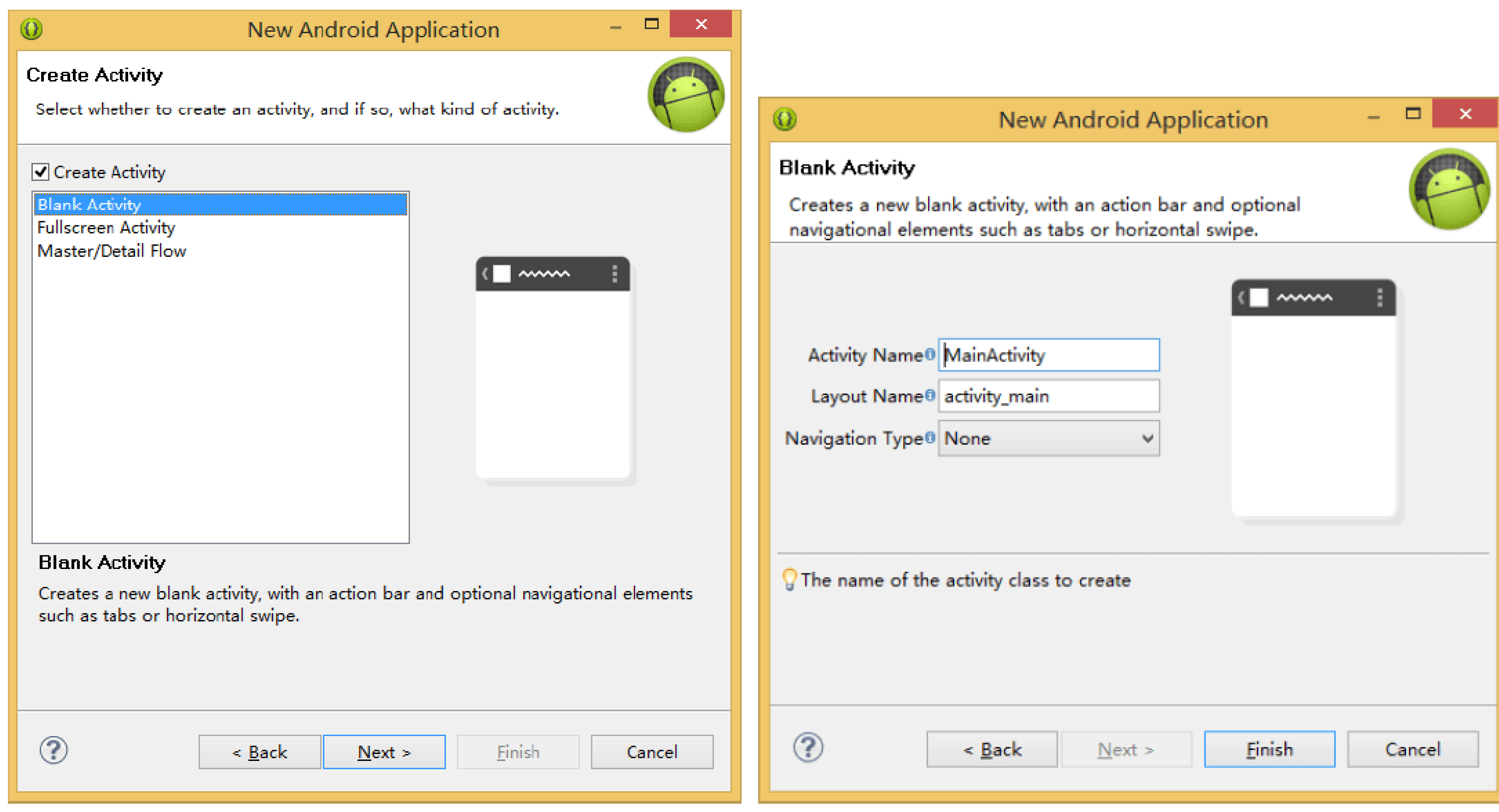


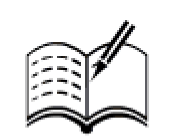
图 2.27　Create Activity 界面　　　　图 2.28　Blank Activity 界面

（8）单击 Finish 按钮，即可创建一个 Android 程序，创建完成的 Android 程序结构如图 2.29 所示。



图 2.29　Android 程序结构





说明：

从图 2.29 中可以看到，res 和 assets 文件夹都用来存放资源文件，但在实际开发时，Android 不为 assets 文件夹下的资源文件生成 ID，用户需要通过 AssetManager 类以文件路径和文件名的方式来访问 assets 文件夹中的文件。

（9）在主 Acitvity 窗口中显示的内容是在 values 目录下的 strings.xml 文件中设置的，打开该文件，将相应的文字内容修改为 Hello Android，其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">2.1</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello Android!</string>
</resources>
```

通过以上步骤即创建了一个显示 Hello Android 的 Android 应用程序。

2.2.2　创建 AVD 模拟器

AVD（Android Virtual Device）即 Android 模拟器，它是 Android 官方提供的一个可以运行 Android 程序的虚拟机，在运行 Android 程序之前，首先需要创建 AVD 模拟器。创建 AVD 模拟器的步骤如下：

（1）启动 Eclipse，单击工具栏中的按钮，或者在菜单栏中依次选择 Window/Android Virtual Device Manager 命令，弹出 Android Virtual Device Manager 窗口，如图 2.30 所示，单击 New 按钮。

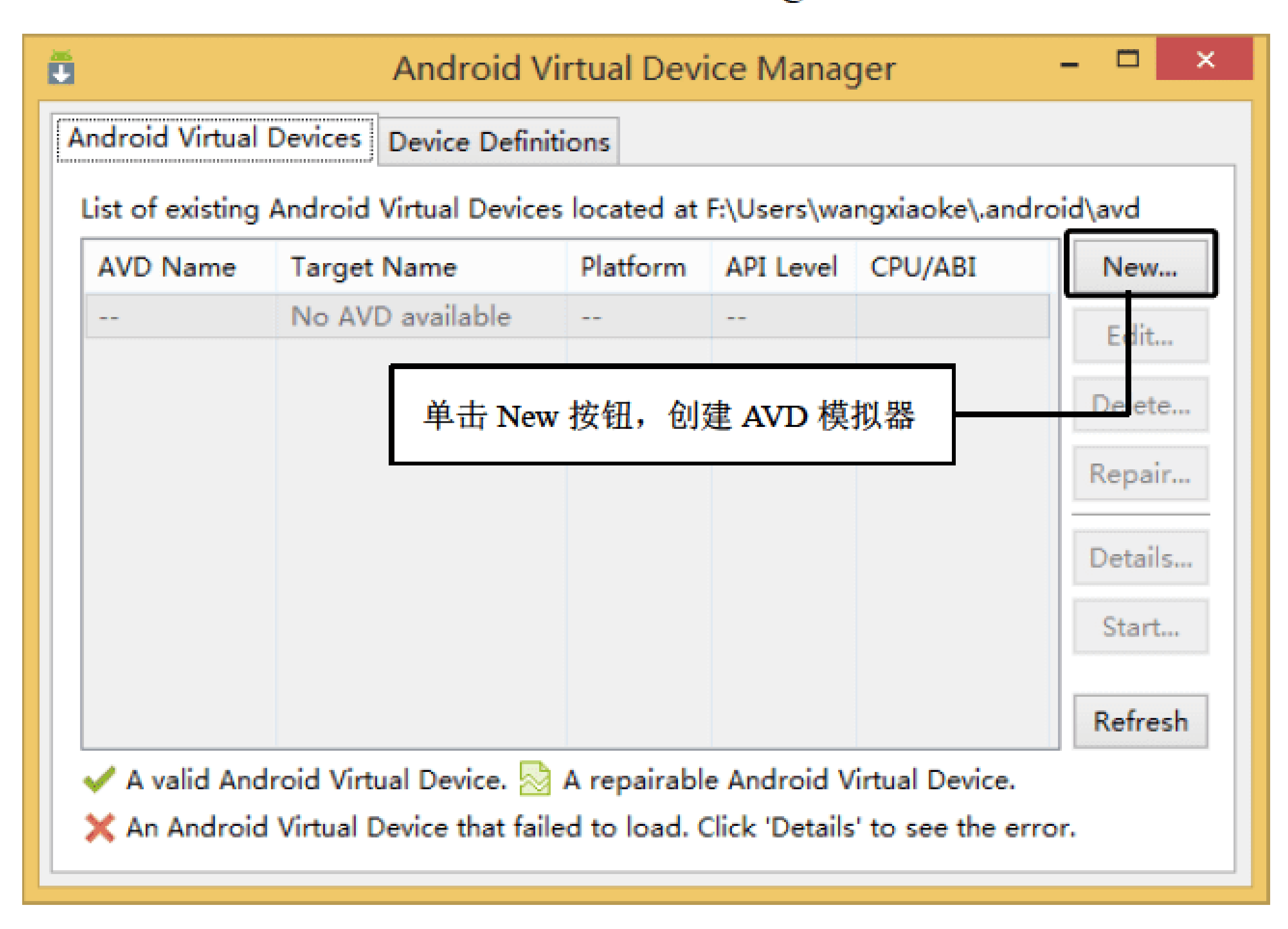
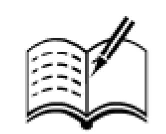


图 2.30　Android Virtual Device Manager 窗口

（2）弹出 Create new Android Virtual Device(AVD)对话框，如图 2.31 所示。在该对话框中，首先输入要创建的 AVD 名称，并选择 AVD 模拟器版本，然后设置 SD 卡的内存大小，并选择屏



幕样式。



Note

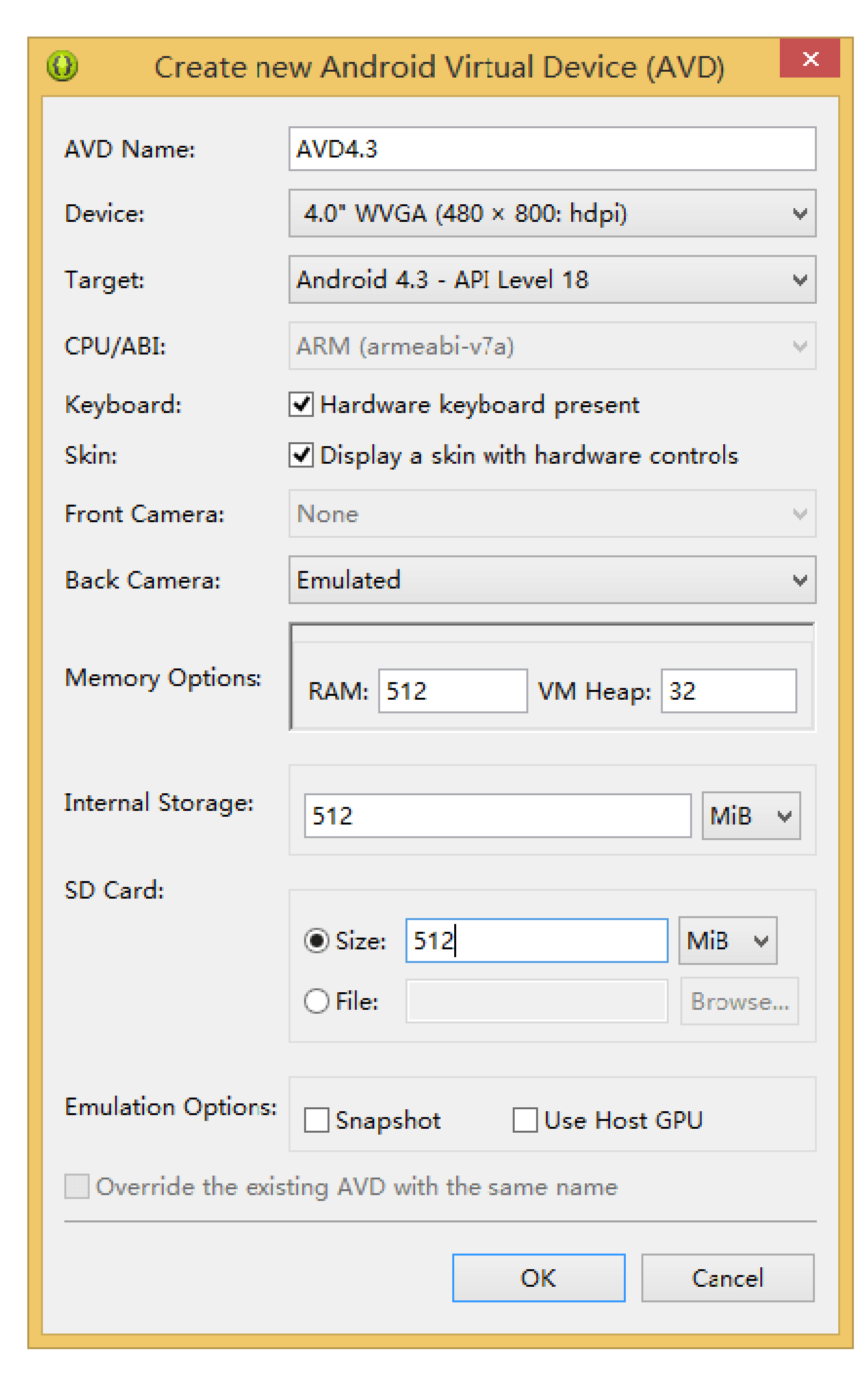


图 2.31　Create new Android Virtual Device(AVD)对话框

注意：

在 AVD Name 文本框中输入 AVD 名称时，中间不能有空格。

（3）单击 OK 按钮，返回 Android Virtual Device Manager 窗口，如图 2.32 所示。这时可以看到已经创建了一个 AVD 模拟器，选中该模拟器，可以通过单击右侧的 Edit、Delete、Details 和 Start 按钮，分别对其进行编辑、删除、查看和启动等操作。

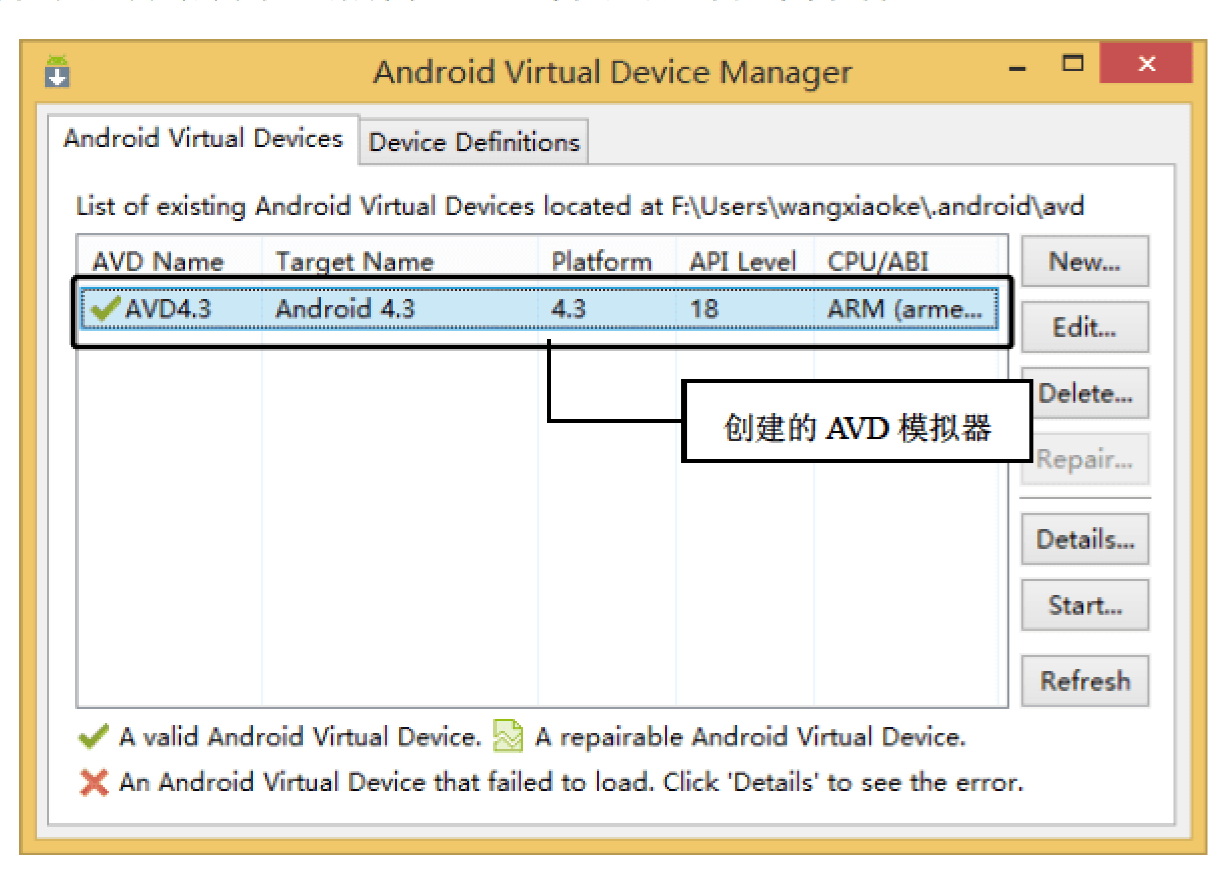


图 2.32　创建完成的 AVD 模拟器





2.2.3　运行 Android 程序

前面两节分别创建了一个 Android 程序和一个 AVD 模拟器，下面来看如何在 AVD 模拟器上运行创建的 Android 程序，其步骤如下：

单击 Eclipse 工具栏中的 按钮，弹出 Run As 窗口，如图 2.33 所示。在该窗口中选择 Android Application 选项，单击 OK 按钮，即可在创建的 AVD 模拟器中运行 Android 程序，运行效果如图 2.34 所示。

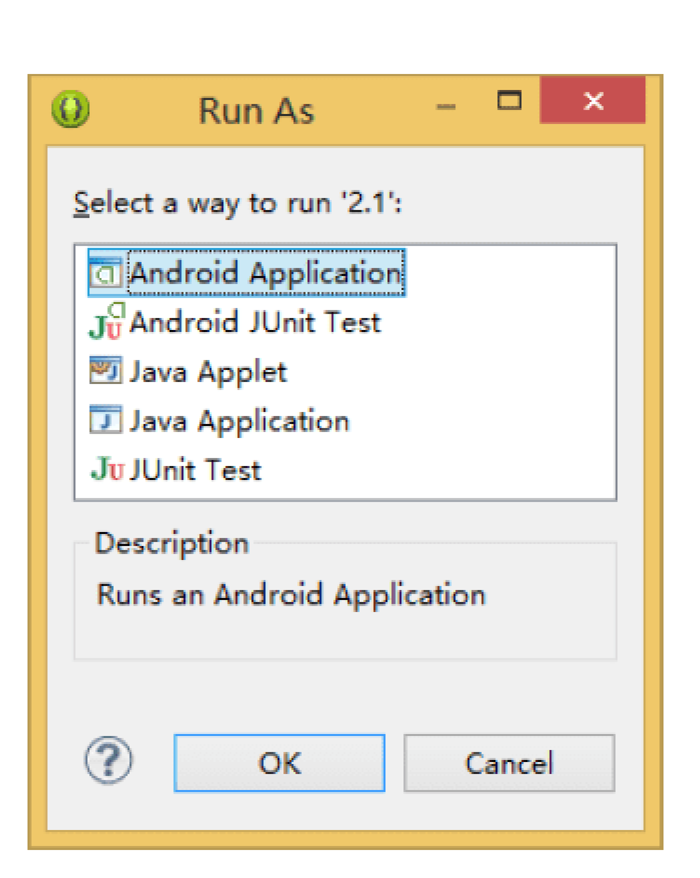


图 2.33　Run As 窗口

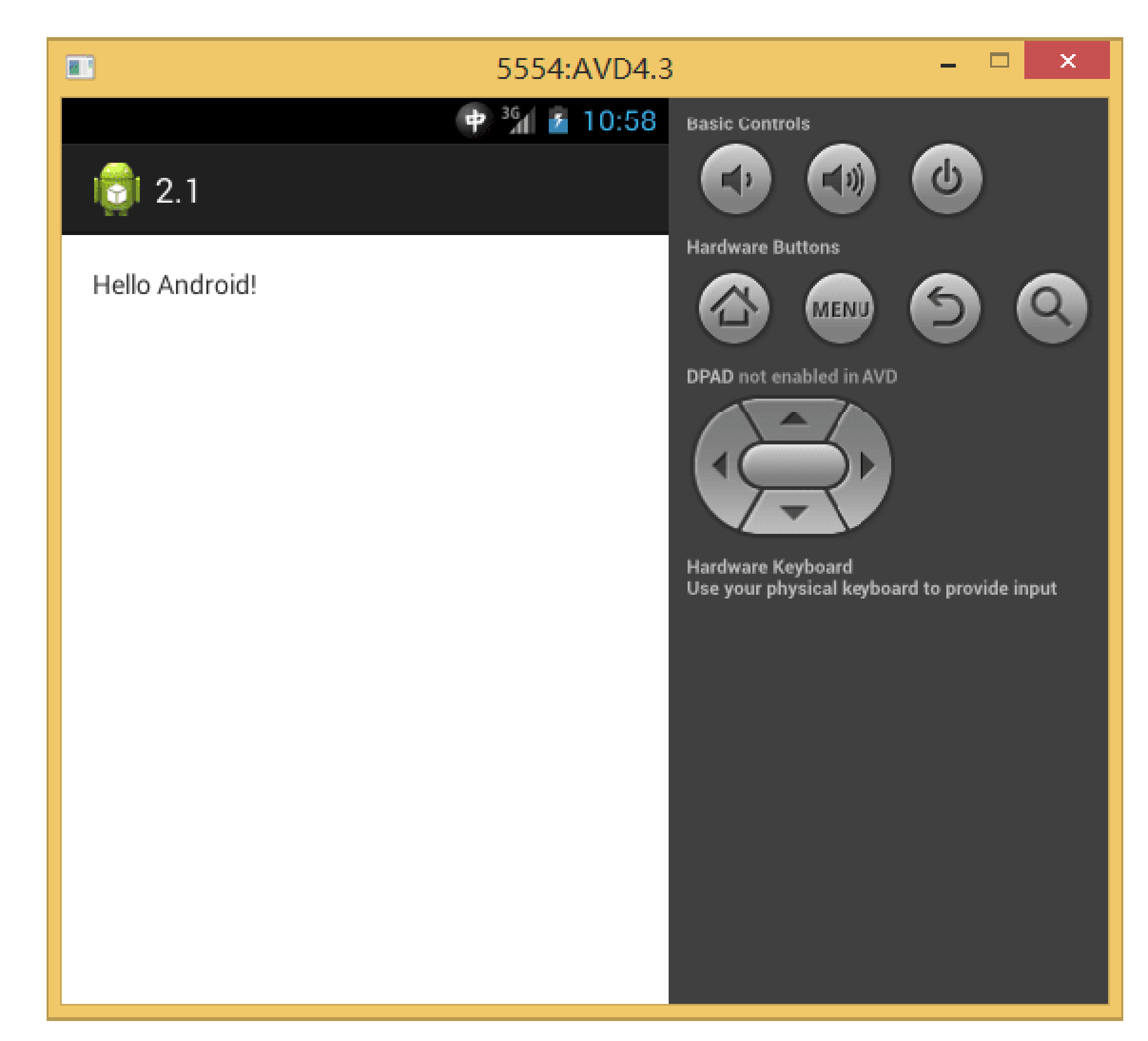
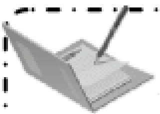


图 2.34　Android 程序运行效果



说明：

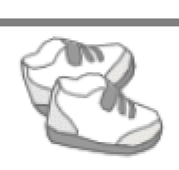
Run As 窗口只在 Android 程序第一次运行时弹出。

2.2.4　调试 Android 应用程序

在开发过程中，肯定会遇到各种各样的问题，这就需要开发人员进行耐心调试。下面先简单了解如何调试 Android 程序。

在 com.xiaoke.helloandroid.activity 包中，有一个名为 MainActivity 的类，将该类的代码替换为如下内容。

```
public class MainActivity extends Activity {
    @SuppressWarnings("null")
        @Override
    public void onCreate(Bundle savedInstanceState) {
```




```
        super.onCreate(savedInstanceState);
        Object object = null;
        object.toString();
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

学习过 Java 语言的读者都可以知道，上面的代码会发生 NullPointerException。启动模拟器后，运行效果如图 2.35 所示。

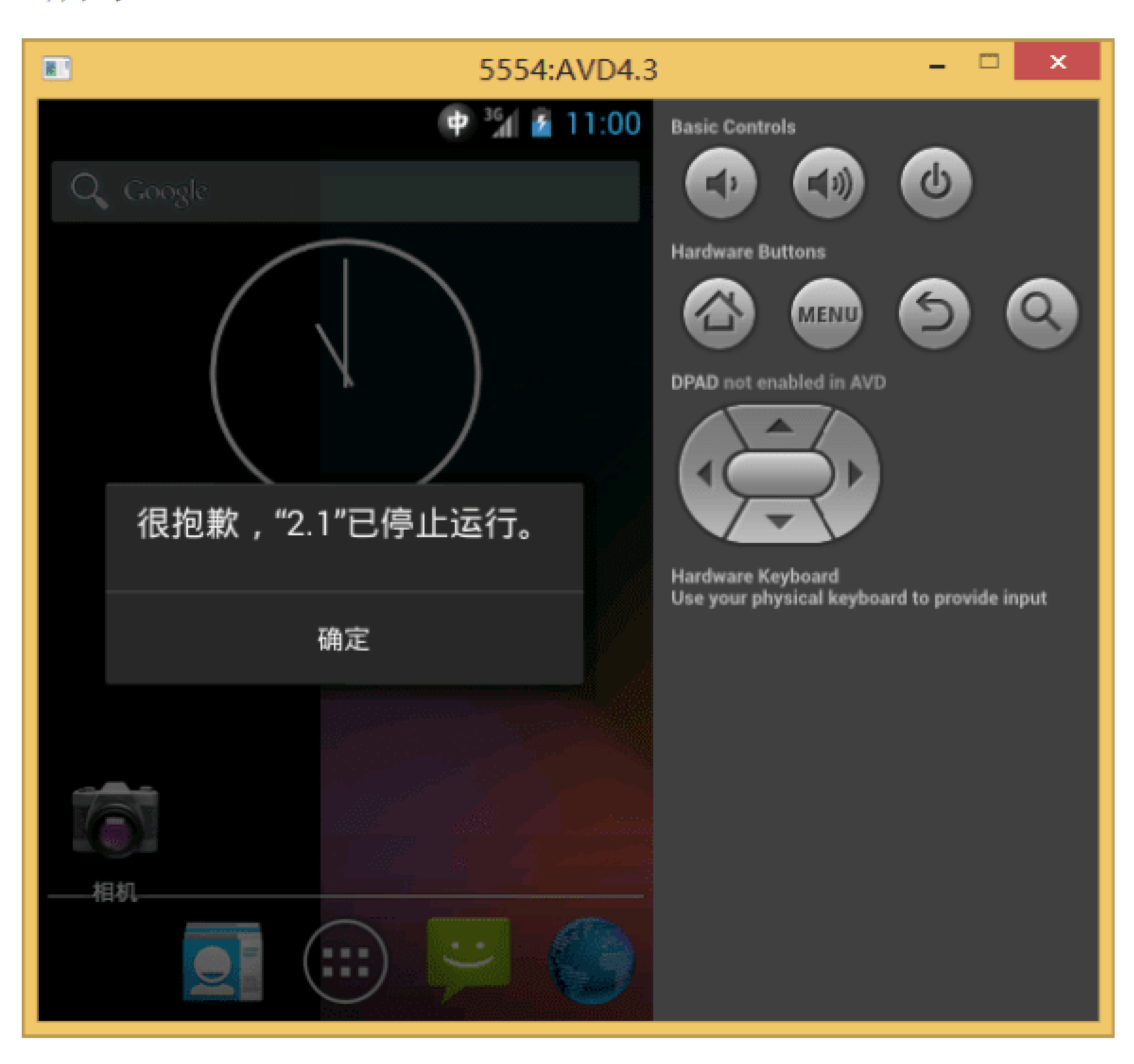


图 2.35　Android 程序出现错误

但是，此时 Eclipse 控制台上并没有提供任何错误信息，那么该如何查看程序到底哪里出现问题了呢？可以使用 LogCat 视图，如图 2.36 所示。其中有一行信息说明 com.xiaoke.helloandroid.activity 包中的 MainActivity 的 onCreate 方法发生了异常。

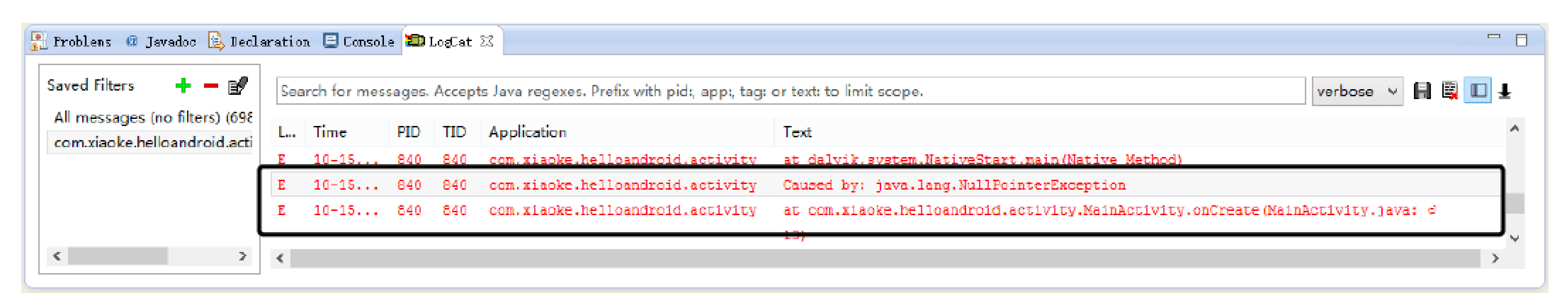
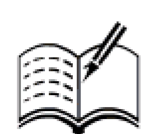


图 2.36　应用程序的异常信息

此处，读者只需要了解：如果程序出现问题，则需要在 LogCat 视图中查找即可。





2.2.5　Android 应用开发流程

前面介绍了如何创建一个 Android 应用，为了加强读者对 Android 开发流程的了解，下面总结一下 Android 程序开发的基本步骤。

☑　创建 Android 虚拟设备或者硬件设备。

开发人员需要创建 Android 虚拟设备（AVD）或者链接硬件设备来安装应用程序。

☑　创建 Android 项目。

Android 项目中包含应用程序使用的全部代码和资源文件。它被构建成可以在 Android 设备安装的.apk 文件。

☑　构建并运行应用程序。

如果使用 Eclipse 开发工具，每次保存修改时都会自动构建，而且可以单击“运行”按钮来安装应用程序到模拟器。如果使用其他 IDE，开发人员可以使用 Ant 工具进行构建，使用 adb 命令进行安装。

☑　使用 SDK 调试和日志工具调试应用。

☑　使用测试框架测试应用程序。

2.3　综 合 应 用

2.3.1　创建一个可以运行在所有 Android 版本上的程序

【例 2.2】 一般的软件对于其开发平台都是向上兼容的，例如对于一个 Android 应用程序来说，如果是在 2.1 版本下开发的，那么它只能运行在 2.1 及其以上版本的 Android 系统中，而不能运行在 2.1 以下版本的 Android 系统中。本实例要求开发一个可以运行在所有 Android 版本上的程序。

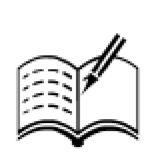
实例位置：光盘\MR\Instance\02\2.2

在本地机器上下载并安装配置 Android 4.3 后，通过 Eclipse 可以创建 Android 4.3 应用程序，在创建过程中，开发人员可以自行选择运行该程序的 Android 最低版本。如果要使创建的 Android 程序能够运行在所有版本的 Android 系统上，只需在创建时，将 Minimum Required SDK 下拉列表框中的 Android 版本设置为最低版本（Android 1.0）即可，如图 2.37 所示。

2.3.2　在 Android 窗口中输出“你好”中文字符串

【例 2.3】 开发一个 Android 应用程序，要求在 Android 窗口中输出一个字体大小为 60dp 的“你好”中文字符串，效果如图 2.38 所示。





Note

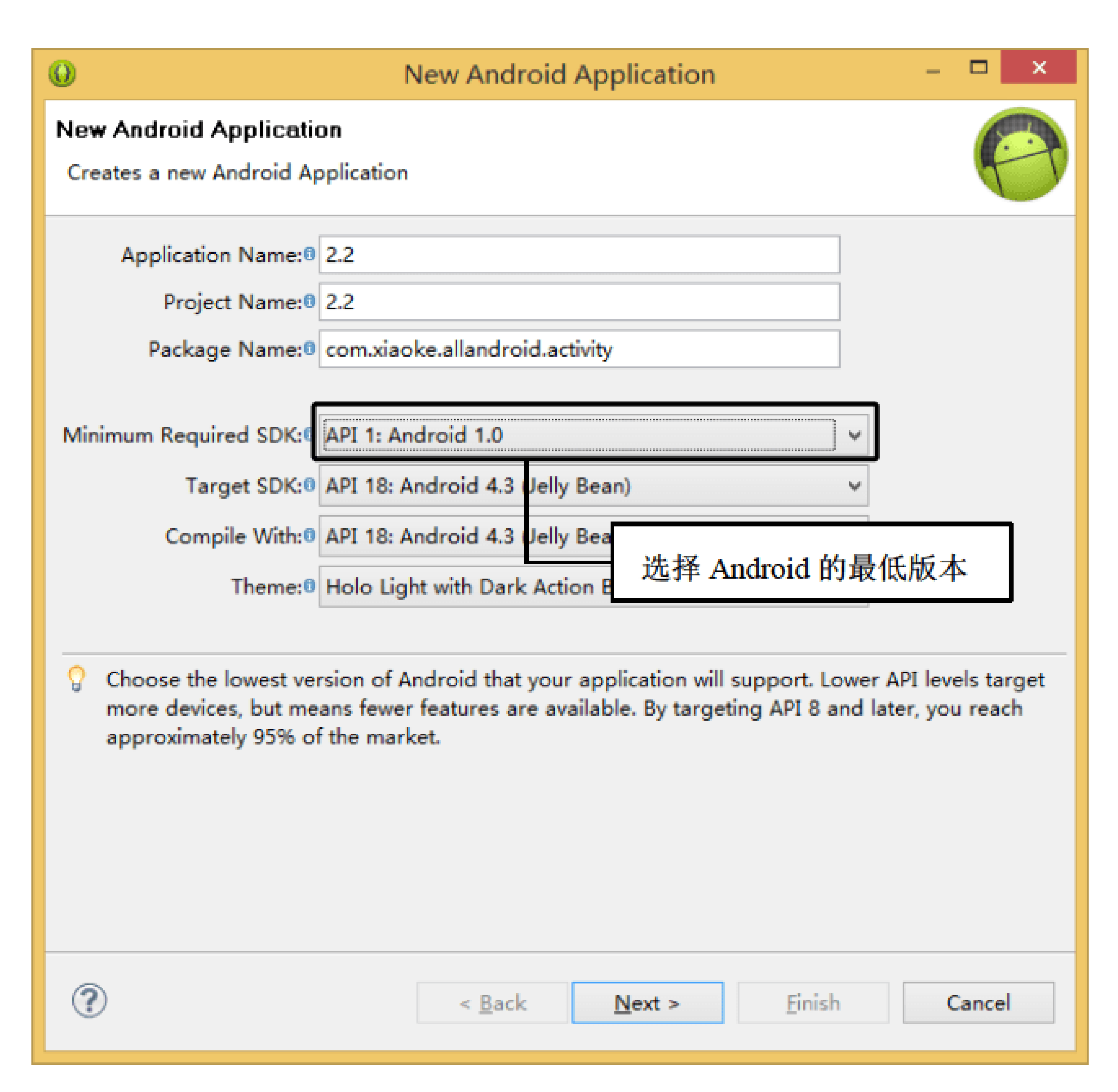


图 2.37　选择 Android 最低版本

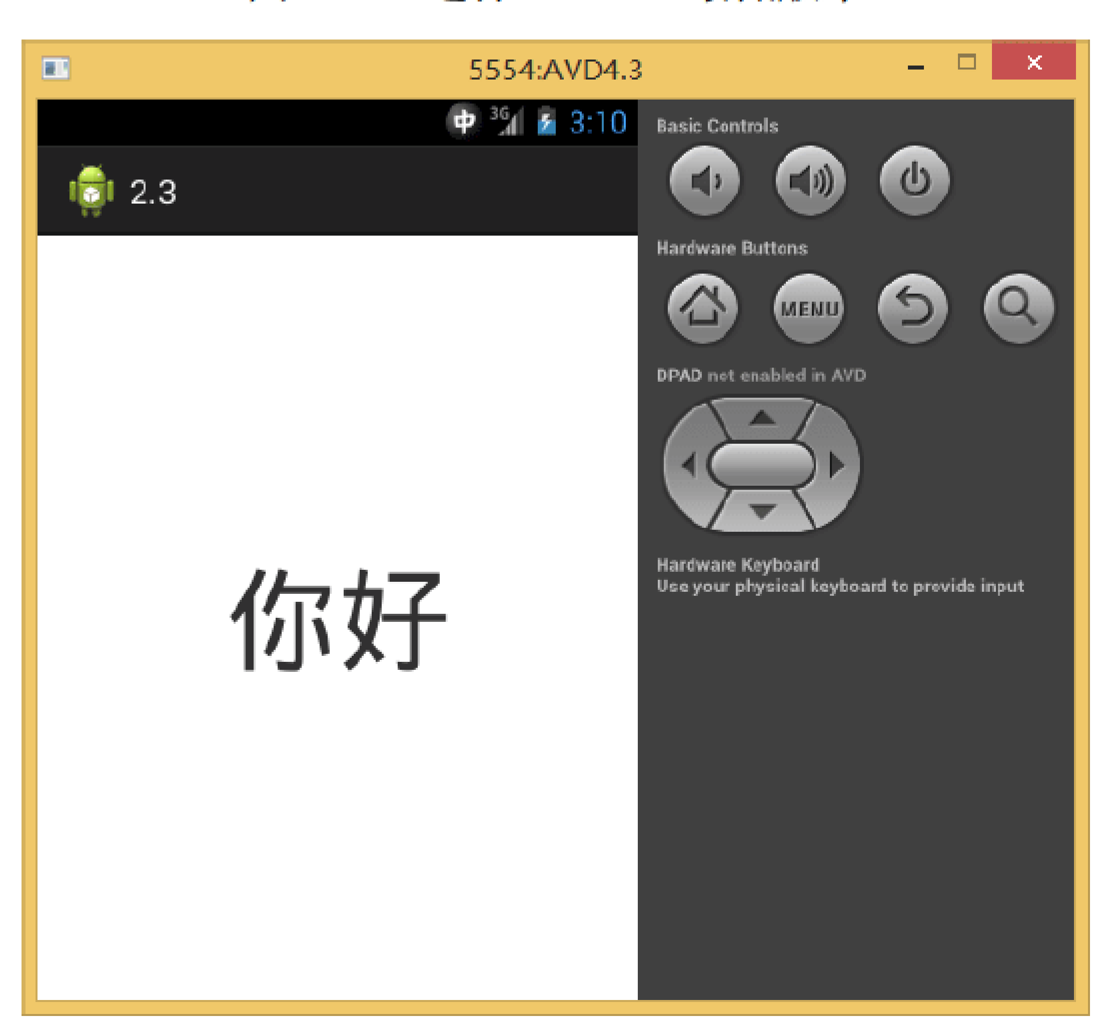


图 2.38　在 Android 窗口中输出“你好”中文字符串

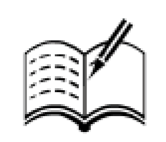
实例位置：光盘\MR\Instance\02\2.3

Android 窗口中的输出内容一般存储在 values 文件夹的 strings.xml 文件中，所以打开该文件，在其中定义要输出的“你好”中文字符串。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">2.3</string>
    <string name="action_settings">Settings</string>
```



```
    <string name="hello_world">你好</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```



在 Android 窗口中默认使用 TextView 控件进行文本的输出，打开主窗口的布局文件 activity_main.xml，在其中设置输出文字的大小和内容。其代码如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:textSize="60dp"
    android:text="@string/hello_world"
    tools:context=".MainActivity" />
```

2.4　本章常见错误

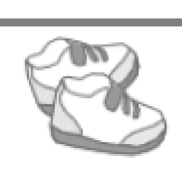
创建并完成一个 Android 程序后，选择 Run As/Android Application 命令，程序并没有在 Android 模拟器中运行，而是出现了如图 2.39 所示的错误提示。

图 2.39　运行 Android 时的错误提示

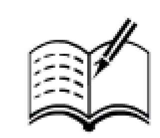
这是由于 Android 模拟器使用超时引起的，Android 模拟器在使用一段时间后会自动超时，从而导致程序无法在 Android 模拟器上体现，遇到这种情况，只需要关闭当前 Android 模拟器，并重新启动即可。

2.5　本 章 小 结

本章主要讲解了在 Windows 系统平台下搭建 Android 开发环境的方法，以及编写、运行 Android 程序的详细步骤。学习本章内容时，搭建 Android 开发环境，编写和运行 Android 程序是本章的重点，读者需要熟练掌握。



2.6 跟 我 上 机



Note

参考答案：光盘\MR\跟我上机

开发一个 Android 程序，将应用程序的名称修改为“Android”。应用程序的名称默认存储在 res\values 文件夹下的 strings.xml 文件中，要修改为应用程序的名称，只需要将<string name="app_name"></string>修改相应的值即可，这里可以将其值修改为“Android”。其代码如下：

```
<string name="app_name">Android</string>
```



认识 Android 模拟器

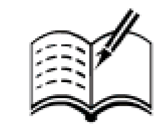
（视频讲解：21 分钟）

Android模拟器是Google官方提供的一款运行Android程序的虚拟机，作为Android开发人员，不管你有没有基于Android操作系统的设备，都可以在Android模拟器上测试自己开发的Android程序。本章将对如何使用Android模拟器进行详细介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 启动指定的 Android 模拟器
- ☐ 删除指定的 Android 模拟器
- ☐ 为 Android 模拟器设置语言和输入法
- ☐ 为 Android 模拟器设置日期时间
- ☐ 使用 adb 命令安装和卸载 Android 程序
- ☐ 通过 DDMS 管理器安装 Android 程序
- ☐ 在 Android 模拟器中卸载程序
- ☐ 设置模拟器桌面背景
- ☐ 在 Android 模拟器中安装搜狗拼音输入法
- ☐ 设置使用 24 小时格式的时间





Note

3.1　启动和删除 Android 模拟器

测试 Android 程序时，一般都是通过 Android 模拟器实现的，本节将介绍如何启动和删除 Android 模拟器。

3.1.1　启动 Android 模拟器

在第 2 章的 2.2.2 节中已经详细讲解了如何创建 Android 模拟器（AVD），本节将详细介绍如何启动 Android 模拟器。启动 Android 模拟器的步骤如下：

（1）单击 Eclipse 工具栏中的按钮，或者在菜单栏中依次选择 Window/Android Virtual Device Manager 命令，弹出 Android Virtual Device Manager 窗口，如图 3.1 所示，在该窗口中选中要启动的 Android 模拟器。

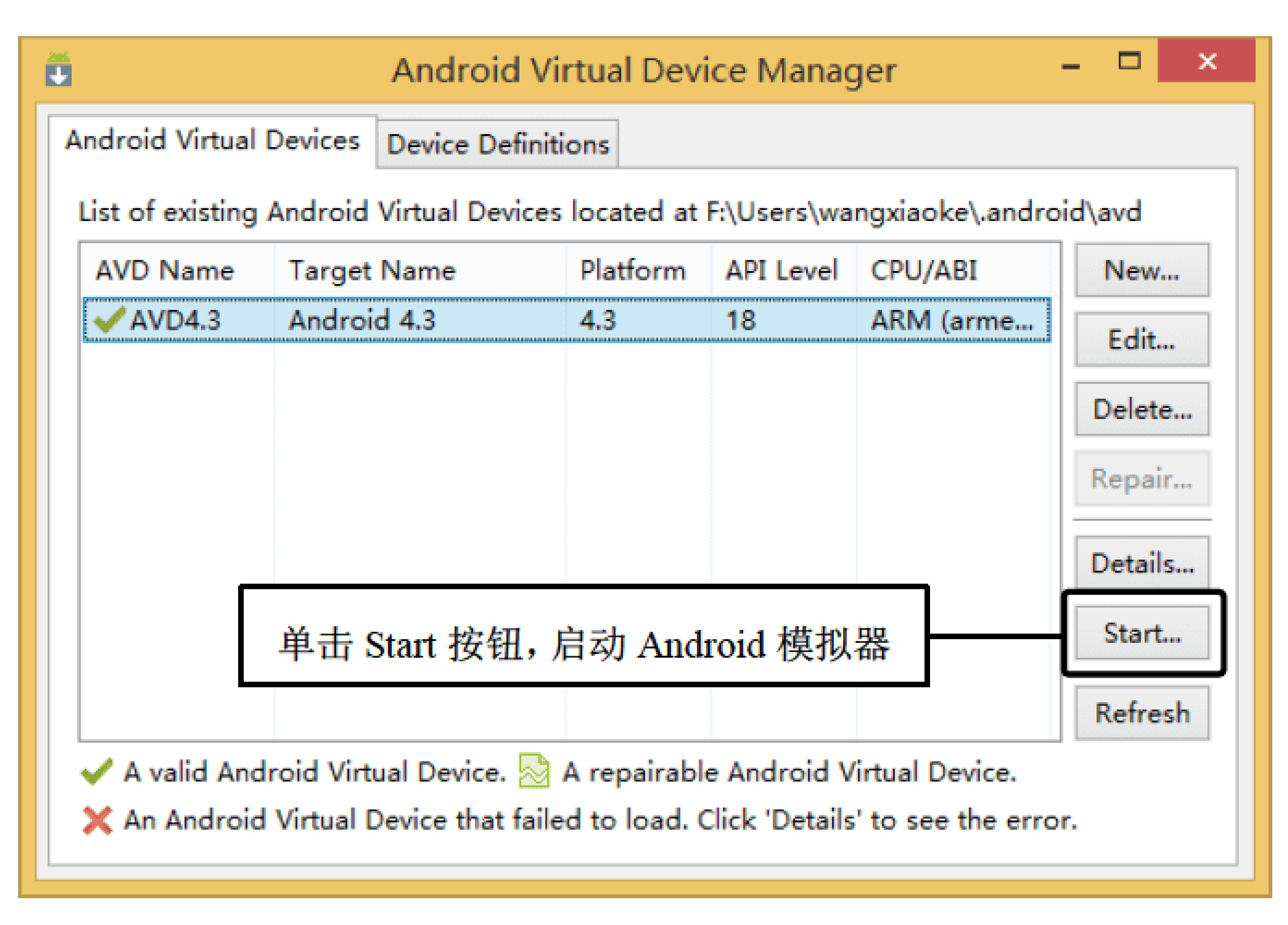


图 3.1　Android Virtual Device Manager 窗口



说明：

在 Android Virtual Device Manager 窗口中可以创建多个 Android 模拟器，但是模拟器的版本和名称不能相同。

（2）单击 Start 按钮，即可启动选中的 Android 模拟器，这里启动的是 4.3 版本的 Android 模拟器，如图 3.2 所示。

（3）从图 3.2 可以看到，Android 模拟器启动后默认处于锁定状态，单击 Android 模拟器中间的“大锁”，会出现如图 3.3 所示的效果，然后将 Android 模拟器中间的“大锁”拖动到右边的“小锁”上，即可解除 Android 模拟器的锁定。



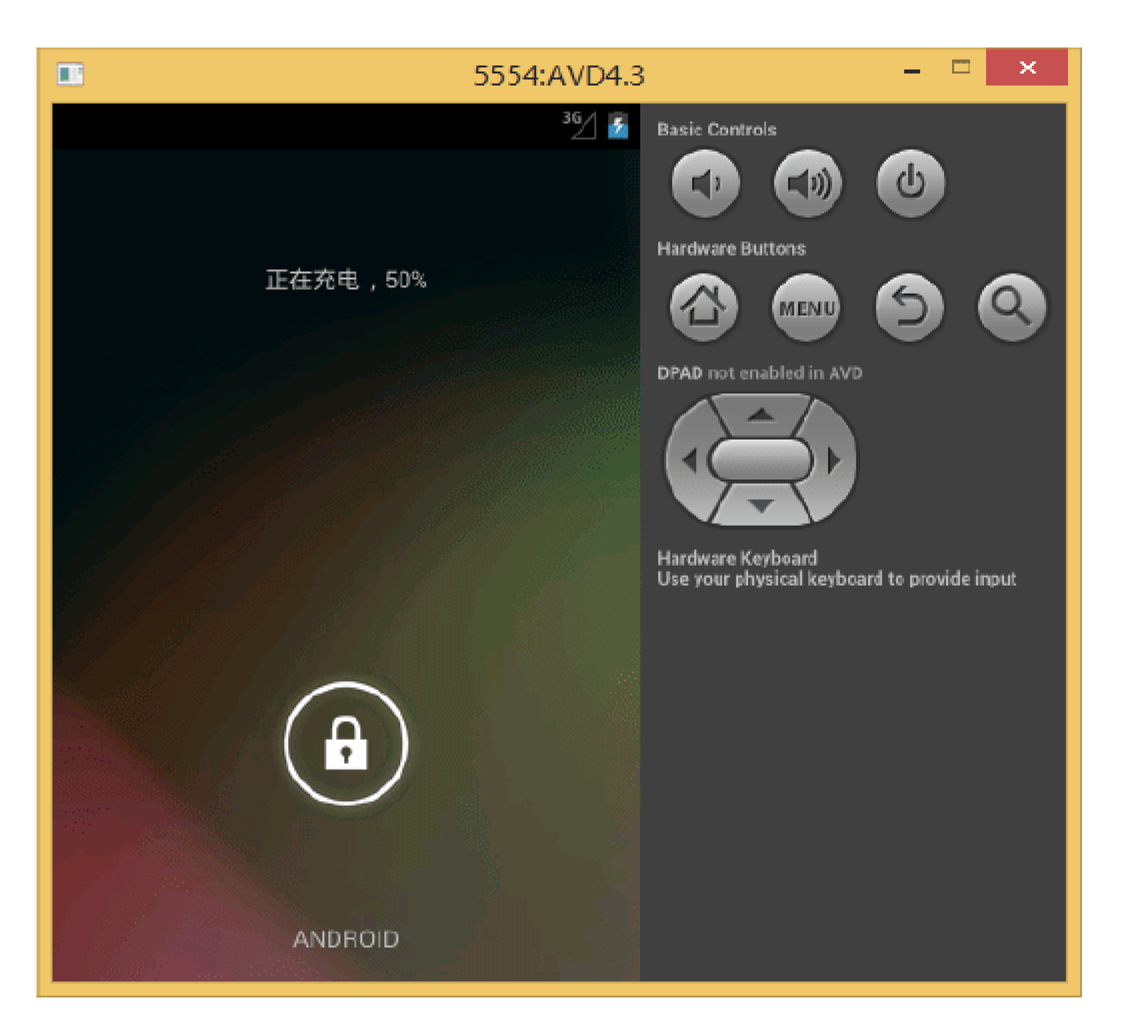


图 3.2　Android 模拟器

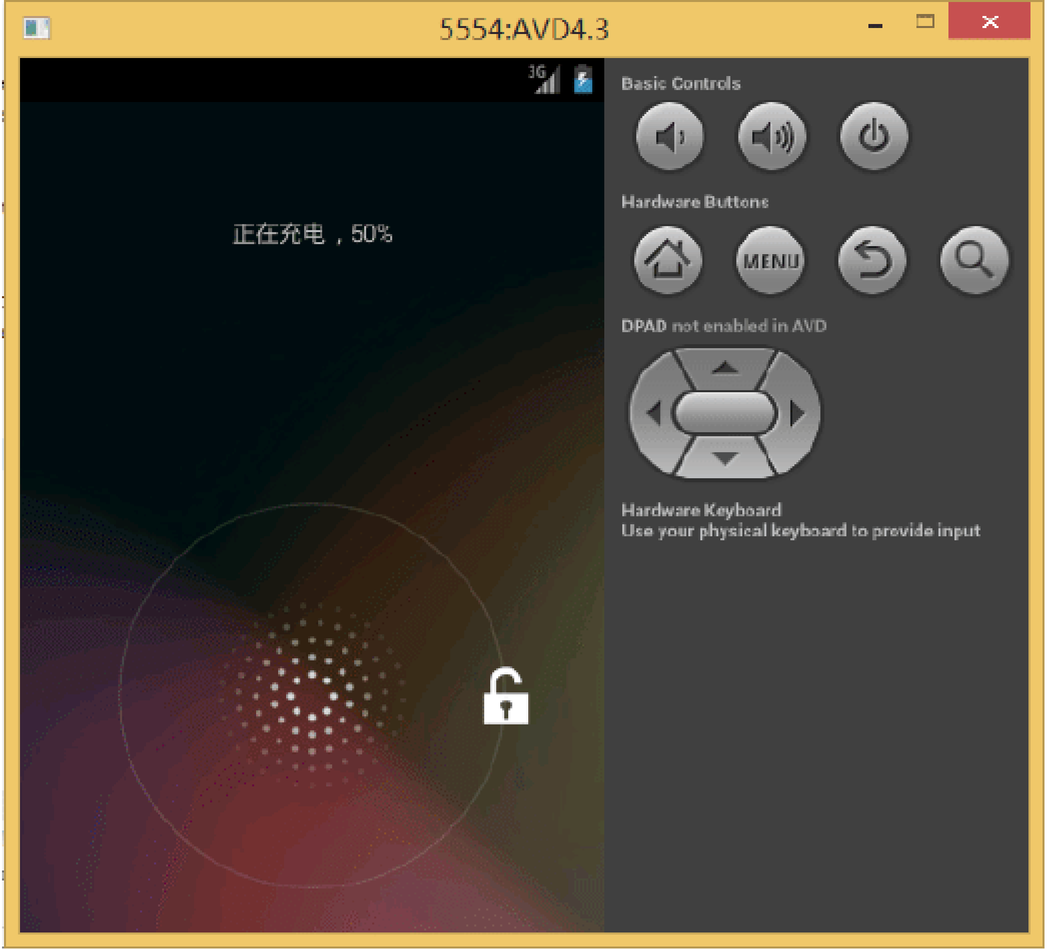


图 3.3　解除 Android 模拟器的锁定状态

3.1.2　删除 Android 模拟器

删除 Android 模拟器的步骤比较简单，只需要在 Android Virtual Device Manager 窗口中选中要删除的 Android 模拟器，然后单击 Delete 按钮即可，如图 3.4 所示。

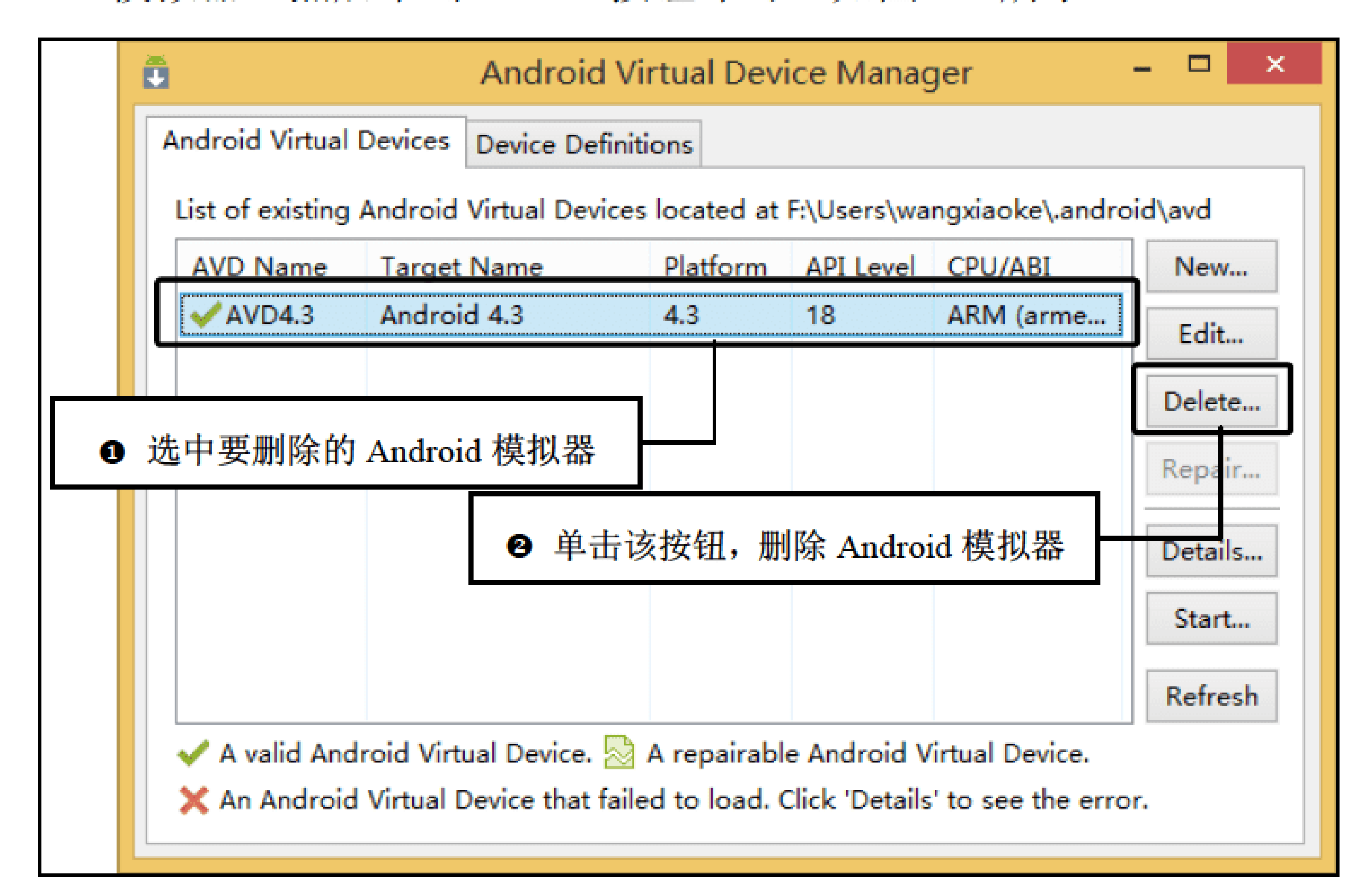


图 3.4　删除 Android 模拟器

3.2　Android 模拟器常用设置

Android 模拟器作为一种基于 Android 操作系统的虚拟设备，它同基于 Android 操作系统的





手机或者平板电脑等设备一样，可以自定义设置。本节将通过语言、输入法和时间等常用设置来熟悉 Android 模拟器。



Note

3.2.1　设置语言

Android 模拟器启动后，默认的语言是英语，为了更方便中国区用户的使用，可以将其默认语言设置为中文，其具体设置步骤如下：

> **注意：**
>
> Android 模拟器的语言可以根据个人所在地域自行设置，如设置为中文（繁体）、Canda（加拿大）等各种语言。

（1）打开 Android 模拟器并解除锁定，如图 3.5 所示。

（2）Android 模拟器第一次使用时，会出现 OK 按钮，单击该按钮，然后单击 Android 主屏最底端的中间按钮，进入 Android 应用程序界面，通过左右翻页找到 Settings 按钮，如图 3.6 所示。



图 3.5　Android 模拟器主屏



图 3.6　Android 应用程序界面

（3）单击 Settings 按钮，进入 Android 模拟器的设置界面并选择 Language & input 选项，如图 3.7 所示。

（4）在打开的列表中选择 Language 选项，如图 3.8 所示。

（5）进入语言选择列表界面，如图 3.9 所示。选择“中文（简体）”选项，这样即可将 Android 模拟器的默认语言设置为中文。





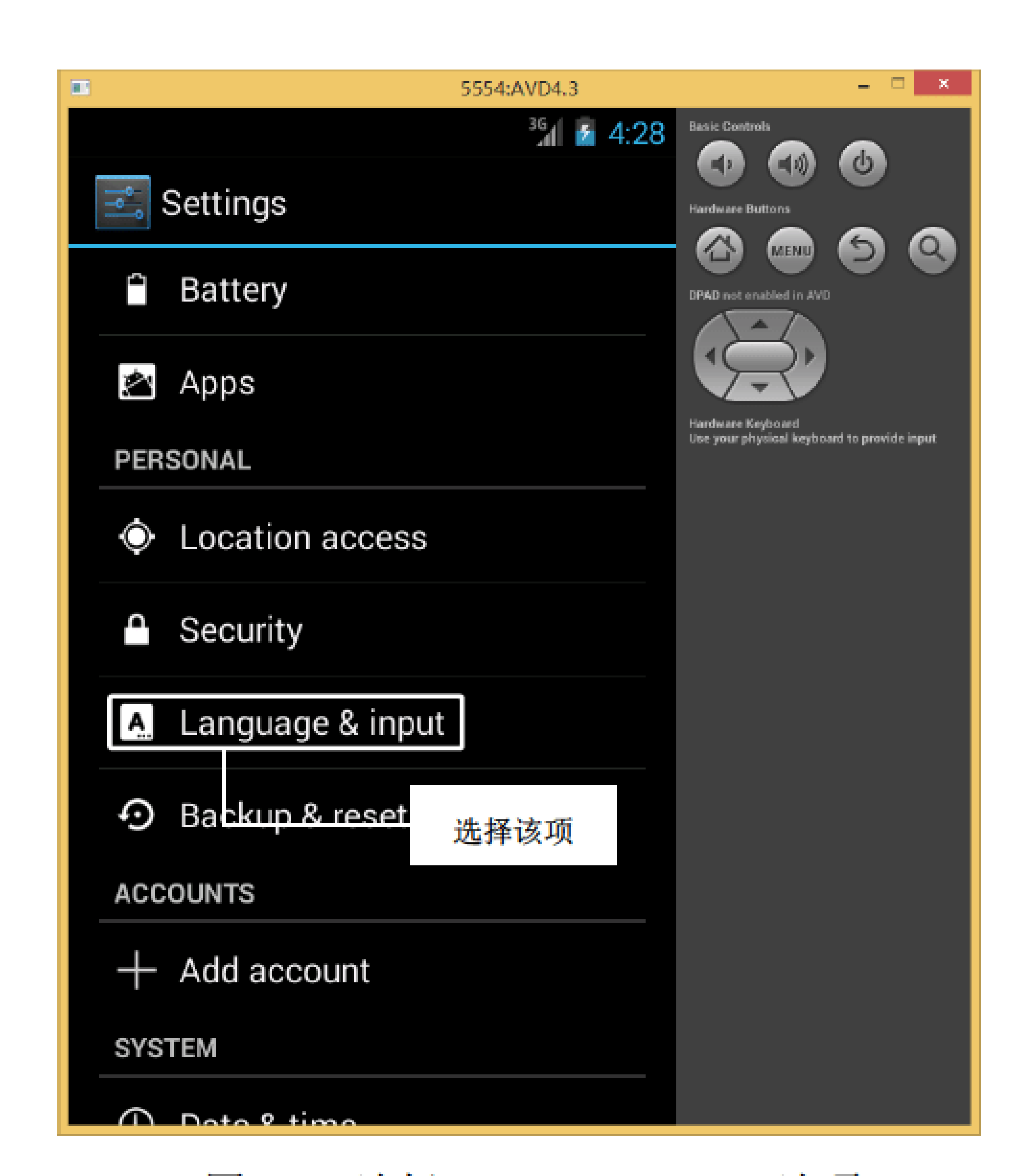


图 3.7　选择 Language & input 选项



图 3.8　选择 Language 选项



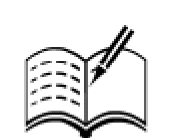
图 3.9　语言选择列表界面

3.2.2　设置输入法

Android 模拟器启动后，默认输入法为 Android 键盘（AOSP），用户可以根据自己的使用习惯对输入法进行设置，这里介绍如何在 Android 模拟器中设置输入法，其具体步骤如下：



（1）在 Android 模拟器的“设置”界面中选择“语言和输入法”选项，如图 3.10 所示。



Note

说明：

由于在 3.2.1 节中已经将 Android 模拟器的默认语言设置为了中文，所以在后面使用 Android 模拟器时，各种菜单的名称都是中文显示的。

（2）在打开的列表中选中“谷歌拼音输入法”复选框，如图 3.11 所示，这样即可将 Android 模拟器的输入法设置为中文输入法。






图 3.10　选择“语言和输入法”选项　　　图 3.11　选中“谷歌拼音输入法”复选框

3.2.3　设置日期时间

Android 模拟器启动后，默认时间为格林尼治时间，这里介绍如何将默认时间设置为中国标准时间，其具体步骤如下：

（1）打开 Android 模拟器，进入其“设置”界面，选择“日期和时间”选项，如图 3.12 所示。

（2）进入“日期和时间”界面，如图 3.13 所示。在界面中，首先将“自动确定日期和时间”和“自动确定时区”两个复选框的选中状态取消，然后单击“选择时区”列表项。

（3）进入“日期和时间——选择时区”界面，在该界面中选择“中国标准时间（北京）”选项，如图 3.14 所示。

（4）返回图 3.13 所示的“日期和时间”界面，在该界面中单击“设置日期”列表项，弹出设置日期对话框，该对话框中设置 Android 模拟器的日期，如图 3.15 所示。



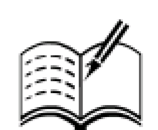

Note



图 3.12 选择“日期和时间”选项

图 3.13 “日期和时间”界面



图 3.14 “日期和时间——选择时区”界面

图 3.15 设置日期

（5）返回图 3.13 所示的“日期和时间”界面，在该界面中单击“设置时间”列表项，弹出“设置时间”对话框，该对话框中设置 Android 模拟器的时间，如图 3.16 所示。

（6）返回图 3.13 所示的“日期和时间”界面，用户还可以通过单击该界面中的“使用 24 小时格式”和“选择日期格式”列表项设置 Android 模拟器的日期和时间格式。通过以上步骤，即可完成 Android 模拟器的日期和时间设置。






Note



图 3.16 “设置时间”对话框

说明：

在设置 Android 模拟器的日期时，可以不手动设置日期，因为在选择了时区后，Android 模拟器会自动获取当前时区的当前日期。

3.3 安装和卸载程序

在 Android 模拟器上安装和卸载程序分别有两种方式：一种是使用 adb 命令安装和卸载 Android 程序；另一种是首先通过 DDMS 管理器安装 Android 程序，然后再通过 Android 模拟器卸载 Android 程序。本节将分别对这两种安装和卸载 Android 程序的方式进行详细讲解。

3.3.1 使用 adb 命令安装和卸载 Android 程序

adb（Android Debug Bridge）是 Android SDK 提供的一个工具，通过该工具可以直接操作 Android 模拟器或者设备，它的主要功能如下：

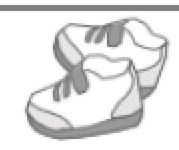
☑ 运行 Android 设备的 shell（命令行）。

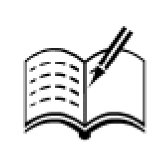
☑ 管理 Android 模拟器或者设备的端口映射。

☑ 在计算机和 Android 设备之间上传或者下载文件。

☑ 将本地 apk 文件安装到 Android 模拟器或者设备上。

下面介绍如何使用 adb 命令安装和卸载 Android 程序。





1. 安装 Android 程序

使用 adb 命令安装 Android 程序的步骤如下：

（1）在“开始”菜单中打开 cmd 命令提示窗口，首先把路径切换到 Android SDK 安装路径的 platform-tools 文件夹，然后使用 adb install 命令将指定的 apk 文件安装到 Android 模拟器上；如果要将 apk 文件安装到 Android 模拟器的 SD 卡上，则使用 adb install –s 命令，如图 3.17 所示。



图 3.17　使用 adb 命令安装 Android 程序

说明：

这里的 apk 文件放在了 Android SDK 安装路径的 platform-tools 文件夹中，所以直接用了 apk 文件名；如果 apk 文件放在其他位置，则需要用它的全路径名。

（2）安装完成后，显示 Success 成功信息，打开 Android 模拟器，可以看到安装的 Android 程序，如图 3.18 所示。



图 3.18　安装的 Android 程序

2. 卸载 Android 程序

使用 adb 命令卸载 Android 程序的步骤如下：

在“开始”菜单中打开 cmd 命令提示窗口，使用 adb uninstall 命令卸载指定的 Android 程序，





如图 3.19 所示。



图 3.19　使用 adb 命令卸载 Android 程序



注意：

使用 adb uninstall 命令卸载 Android 程序时，后面跟的是该程序的包名，而不是 apk 安装文件名。

3.3.2　通过 DDMS 管理器安装 Android 程序

DDMS（Dalvik Debug Monitor Service）是 Android 开发环境的 Dalvik 虚拟机调试监管服务，使用它，可以很方便地为 Android 模拟器安装 Android 程序，其具体步骤如下：

说明：

在 Eclipse 集成开发环境中提供了 DDMS 管理器窗口，如果没有，开发人员可以通过 Android SDK 安装路径下的 tools 文件夹中的 ddms.bat 文件打开。

（1）启动 Eclipse，在其工具栏中单击 DDMS 按钮，切换到“DDMS 管理器”窗口，如图 3.20 所示。在该窗口中，依次展开 data/app 节点，并选中 app 节点，单击按钮。

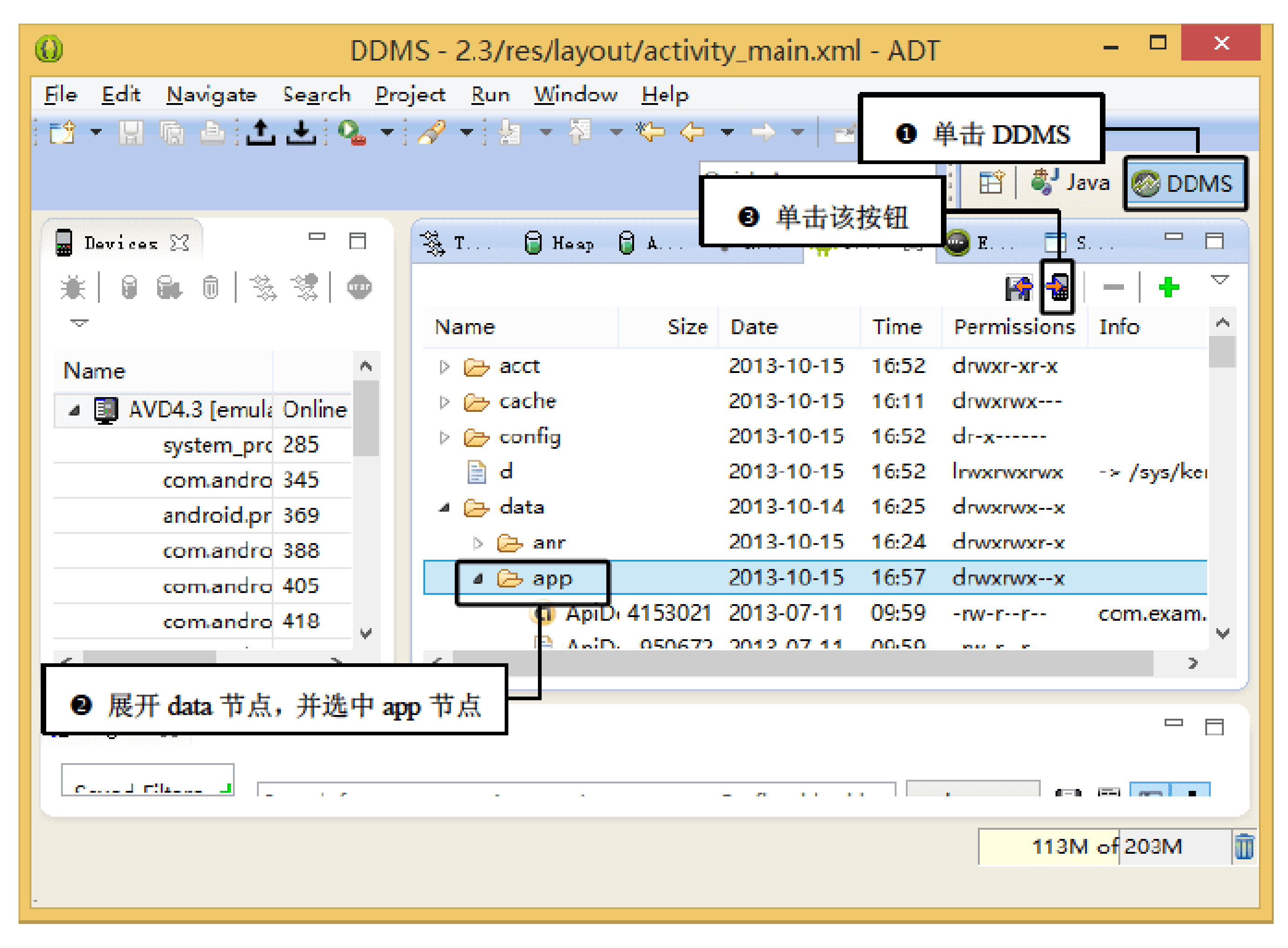


图 3.20　“DDMS 管理器”窗口

（2）打开如图 3.21 所示的 Put File on Device 对话框，在该对话框中选中要安装的 Android





Note

程序所对应的 apk 文件，单击“打开”按钮，即可将 Android 程序安装到 Android 模拟器上。



图 3.21　Put File on Device 对话框



技巧：

用户也可以将 Android 模拟器中自带的 Android 程序的 apk 文件下载到本地机器上，具体操作时，只需要在“DDMS 管理器”窗口的 app 节点下选中指定的 Android 程序，然后单击 按钮即可。

3.3.3　在 Android 模拟器中卸载程序

前面讲解了使用 adb uninstall 命令卸载 Android 模拟器中的 Android 程序，那么可不可以直接在 Android 模拟器中卸载 Android 程序呢？答案是肯定的。本节将详细介绍如何在 Android 模拟器中卸载 Android 程序，其具体步骤如下：

（1）打开 Android 模拟器的“设置”界面，在列表中选择“应用”选项，如图 3.22 所示。

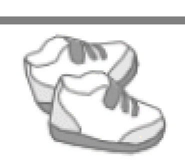
（2）进入“应用”界面，如图 3.23 所示。



图 3.22　选择“应用”选项



图 3.23　“应用”界面





Note

（3）在图 3.23 中选择要卸载的 Android 程序，进入“应用信息”界面，如图 3.24 所示。在该界面中，首先单击“强行停止”按钮，停止 Android 程序的运行，然后单击“卸载”按钮，即可卸载指定的 Android 程序。



图 3.24 “应用信息”界面

3.4 综 合 应 用

3.4.1 设置模拟器桌面背景

在 Android 4.3 的模拟器中，提供了多种桌面背景，下面讲解如何进行设置，其具体步骤如下：

（1）启动模拟器，进入设置列表，在设置列表中，找到“显示”列表项，如图 3.25 所示。

（2）单击“显示”列表项，将进入“显示”界面，在该界面中，找到“壁纸”列表项，如图 3.26 所示。



图 3.25 Android 4.3 模拟器设置界面



图 3.26 Android 4.3 模拟器显示设置界面





（3）单击“壁纸”列表项，将显示“选择壁纸来源”界面，如图 3.27 所示。

（4）单击“壁纸”列表项，将进入到如图 3.28 所示的界面。在该界面中，滑动下方的画廊视图可以切换不同的背景图片，当前居中显示的背景图片会显示预览效果。单击“设置壁纸”按钮完成设置。



图 3.27　Android 4.3 模拟器墙纸设置界面

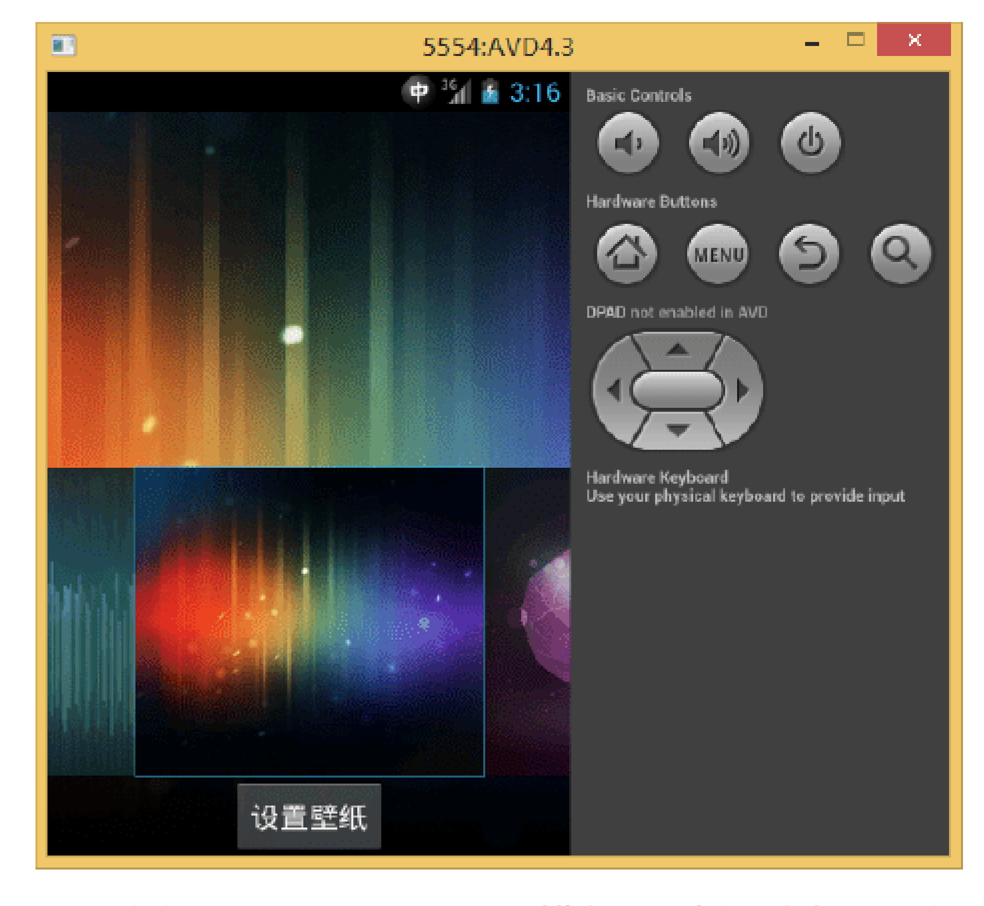


图 3.28　Android 4.3 模拟器壁纸选择界面

3.4.2　在 Android 模拟器中安装搜狗拼音输入法

为了便于输入中文，下面讲解如何安装搜狗拼音输入法，其具体步骤如下：

（1）在搜狗拼音输入法官方网站 pinyin.sogou.com 下载 Android 版本的安装包，当前版本的文件名是 SogouInput_android_1.6.5_sweb.apk，将下载好的文件保存到 Android SDK 的 platform-tools 文件夹下。

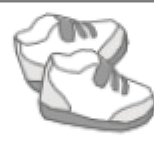
（2）运行 Android 模拟器，启动控制台，输入“adb install SogouInput_android_1.6.5_sweb.apk”命令进行安装，如图 3.29 所示。



图 3.29　安装搜狗拼音输入法

（3）安装完成后，在应用程序界面会显示搜狗拼音输入法的图标，如图 3.30 所示。

（4）单击“搜狗输入法”图标，进入输入法设置界面，如图 3.31 所示，根据提示完成对输入法的设置。

Note

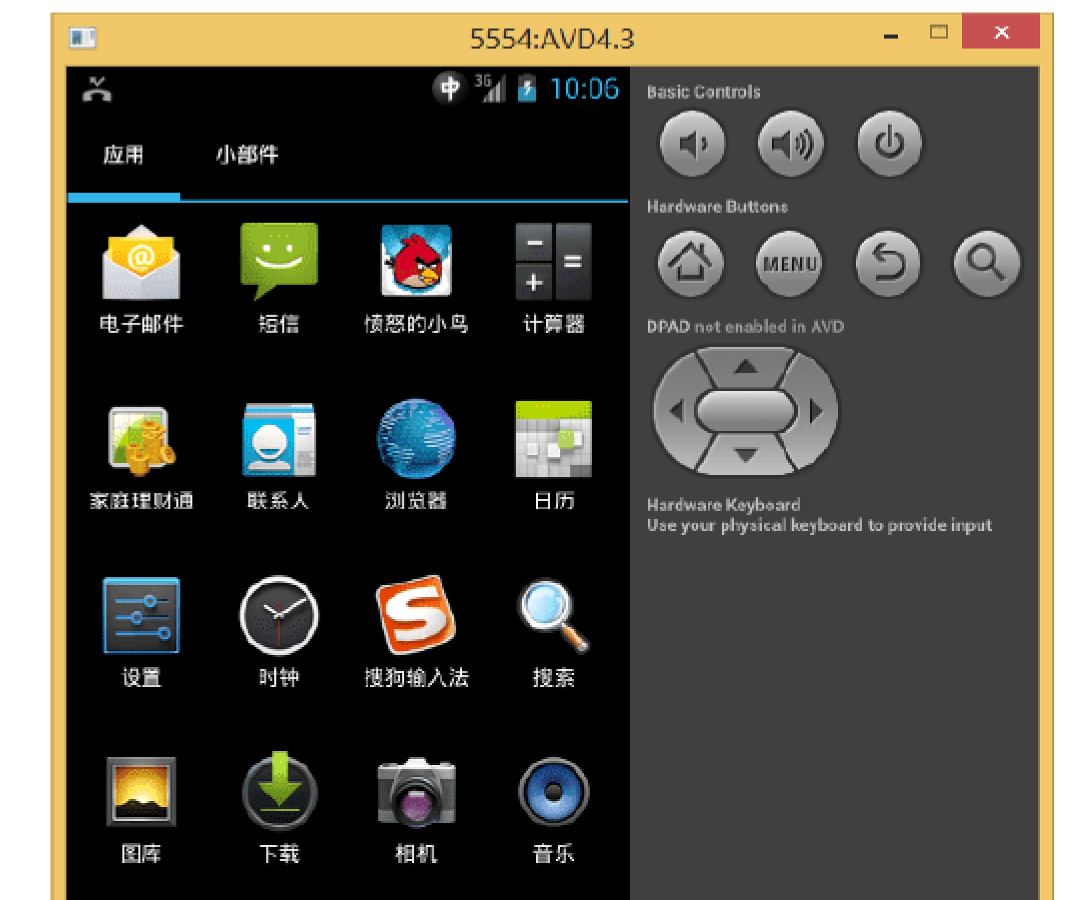



图 3.30　模拟器应用程序界面



图 3.31　搜狗输入法设置界面

3.5　本章常见错误

使用 adb install 命令在 cmd 窗口中安装 apk 文件时，出现如图 3.32 所示的错误提示。

```
F:\WINDOWS\system32\cmd.exe - adb install SogouInput.apk
Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。

F:\Users\wangxiaoke>D:

D:\>cd D:\安装文件\JAVA安装工具\adt-bundle-windows-x86-20130917\sdk\platform-too
ls

D:\安装文件\JAVA安装工具\adt-bundle-windows-x86-20130917\sdk\platform-tools>adb
install SogouInput.apk
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
error: device not found
- waiting for device -
```

图 3.32　使用 adb install 命令安装 apk 文件时的错误提示

出现这种错误提示主要是由于没有找到 Android 设备引起的。在使用 adb install 命令安装 apk 文件时，一定要确保已经启动了 Android 模拟器或者其他的 Android 设备。

3.6　本 章 小 结

本章主要对 Android 模拟器的使用进行了详细讲解，主要包括 Android 模拟器的创建、启动、删除以及一些常用功能（如语言、输入法和日期时间等）的设置，另外还重点讲解了如何在 Android 模拟器上安装和卸载程序。通过本章的学习，读者应该能够熟悉 Android 模拟器的使用。管理 Android 模拟器和在 Android 模拟器上安装、卸载程序是本章的重点，读者应该熟练掌握。





3.7　跟 我 上 机

参考答案：光盘\MR\跟我上机

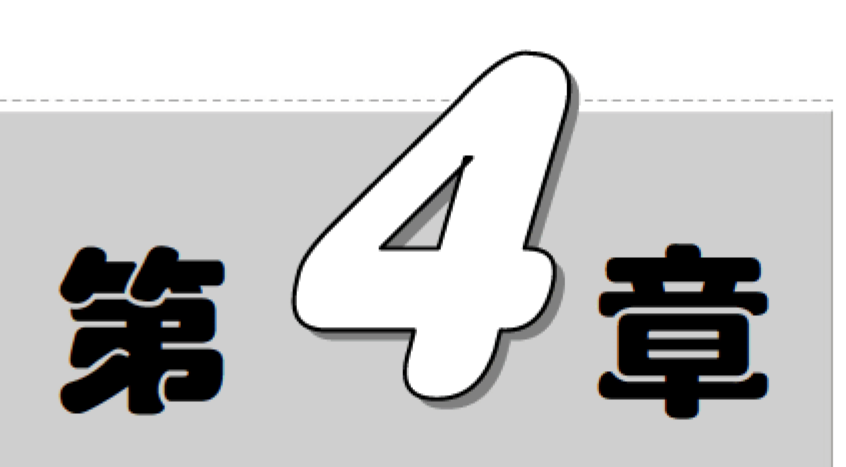
Android 模拟器启动后，默认时间为 12 小时格式的时间，即通过 AM 和 PM 来表示上午、下午的时间。实际上，它还提供了另一种采用 24 小时格式的时间。下面将介绍如何设置使用 24 小时格式的时间，其具体步骤如下：

（1）打开 Android 模拟器，进入其设置界面，选择“日期和时间”选项。

（2）在进入的“日期和时间”界面中，选中“使用 24 小时格式”复选框，如图 3.33 所示，即可设置为使用 24 小时格式的时间。



图 3.33　选中“使用 24 小时格式”复选框



剖析 Android 程序

（视频讲解：58 分钟）

在开发 Android 程序之前，首先需要对 Android 程序设计的基础进行充分的了解，本章将对 Android 程序的结构、生命周期及基本组件等基础内容进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- □ Android 程序的组成及结构
- □ Android 程序的生命周期
- □ 改变默认布局文件的背景图片并在 Activity 中显示
- □ 创建一个广播接收器
- □ 获取 Android 系统中存储的联系人编号及姓名
- □ 查看 Android 模拟器中正在运行的服务
- □ 在 Android 程序中添加 Activity
- □ 在 Android 程序中添加 Service
- □ 添加名称为 com.mingrisoft02 的包





4.1　Android 程序的组成

第 2 章的 2.2.1 节创建了一个 Hello Android 程序，其代码是由 ADT 插件自动生成的，所以当时没有对其结构进行分析，本节借用 Hello Android 程序的结构对 Android 程序的结构进行详细介绍。Hello Android 程序的结构如图 4.1 所示。



图 4.1　Hello Android 程序的结构

从图 4.1 中可以看出，一个 Android 程序中包含很多目录，下面对比较重要的目录进行详细介绍。

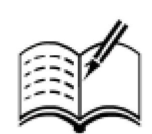
4.1.1　src 目录

src 目录包含了 Android 程序的所有包及源文件（.java），如图 4.1 中的 src 目录中包含了 com.xiaoke.helloandroid.activity 包和 MainActivity.java 源文件，如图 4.2 所示。



图 4.2　src 目录中包含的包及源文件





Note

例如，下面看一下 MainActivity.java 源文件的初始代码，其代码如下：

```
package com.xiaoke.helloandroid.activity;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
public class MainActivity extends Activity {
    @SuppressWarnings("null")
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

从上面的代码可以看出，开发人员创建的 MainActivity 类默认是继承自 Activity 类的，并且在该类中，重写了 Activity 类中的 onCreate()方法，在 onCreate()方法中通过 setContentView (R.layout.activity_main)设置当前的 Activity 要显示的布局文件。



说明：

这里使用 R.layout.activity_main 来获取 layout 文件夹中的 activity_main.xml 布局文件，这是因为，在 Android 程序中，每个资源都会在 R.java 文件中生成一个索引，而通过这个索引，开发人员可以很方便地调用 Android 程序中的资源文件。

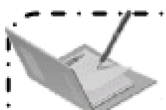
4.1.2 res 目录

res 目录中包含了 Android 程序中的所有资源，如程序图片、布局文件和常量值等。例如，图 4.1 中的 res 目录中包含了 Hello Android 程序的图标文件、布局文件和常量值，其中，drawable-hdpi、drawable-ldpi、drawable-mdpi、drawable-xhdpi 和 drawable-xxhdpi 这 5 个文件夹中存储程序图片文件，layout 文件夹中存储布局文件，values 文件夹中存储常量值等，如图 4.3 所示。



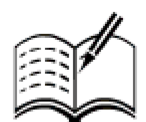
图 4.3 res 目录中包含的资源





说明：

使用 Android 2.3 之前的版本创建 Android 程序时，程序结构中只有一个 drawable 文件夹，用来存储程序图片文件。





Note

下面对 res 目录中的几个主要文件夹及其文件进行详细介绍。

1．程序图片文件夹

在 Android 4.3 版本中，Android 程序中的程序图标文件夹默认有 drawable-hdpi、drawable-ldpi、drawable-mdpi、drawable-xhdpi 和 drawable-xxhdpi 5 个，在向其中添加图标或者图片资源时，只需要在 Eclipse 中选中这 5 个文件夹中的任意一个，然后将要添加的图标或图片资源复制到其中即可。

2．layout 文件夹

layout 文件夹主要用来存储 Android 程序中的布局文件，在创建 Android 程序时，会默认生成一个 main.xml 布局文件。

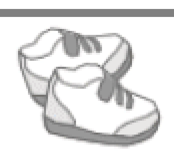
例如，activity_main.xml 布局文件的默认代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />
</RelativeLayout>
```

activity_main.xml 布局文件中的重要元素及说明如表 4.1 所示。

表 4.1　activity_main.xml 布局文件中的重要元素及说明

元　素	说　明
RelativeLayout	布局元素
xmlns:android	包含命名空间的声明，其属性值为 http://schemas.android.com/apk/res/android，表示 Android 中的各种标准属性能在该 xml 文件中使用，它提供了大部分元素中的数据
xmlns:tools	指定布局的默认工具
android:layout_width	指定当前视图在屏幕上所占的宽度
android:layout_height	指定当前视图在屏幕上所占的高度
TextView	TextView 元素，用来显示文本
android:text	TextView 的文本值




Note

技巧：

开发人员在指定各个元素的属性值时，可以按下 Alt+? 快捷键来显示帮助列表，然后在帮助列表中选择系统提供的值，如图 4.4 所示。

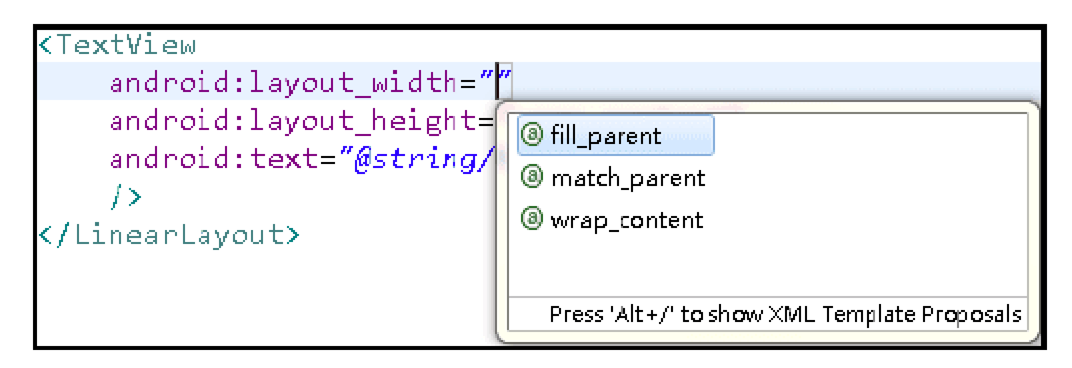


图 4.4　按下 Alt+? 快捷键来显示帮助列表

3．values 文件夹

values 文件夹是 Android 程序中的全局数据文件夹，在创建 Android 程序时，会默认生成一个 strings.xml 数据文件。

例如，strings.xml 数据文件的默认代码如下：

```
<resources>
    <string name="app_name">Hello Android</string>
    <string name="hello_world">Hello Android!</string>
    <string name="menu_settings">Settings</string>
    <string name="title_activity_main">Hello Android</string>
</resources>
```

从上面的代码可以看出，strings.xml 文件中默认存在一个<resources>节点，该节点用来标明其子节点元素是资源，然后看到默认生成了 4 个字符串资源。

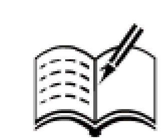
4.1.3　gen 目录及 R.java 文件

gen 目录是 Android 程序中的一个特殊目录，该目录中有一个 R.java 文件，该文件是创建 Android 程序时自动生成的。R.java 文件用来定义 Android 程序中所有资源的索引，在 java 源文件中编写代码时，可以直接通过该索引访问各种资源。

例如，Hello Android 程序中的 R.java 文件代码如下：

```
package com.xiaoke.helloandroid.activity;
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        public static final int padding_large=0x7f040002;
        public static final int padding_medium=0x7f040001;
        public static final int padding_small=0x7f040000;
    }
```






```
        public static final class drawable {
            public static final int ic_action_search=0x7f020000;
            public static final int ic_launcher=0x7f020001;
        }
        public static final class id {
            public static final int menu_settings=0x7f080001;
            public static final int textView1=0x7f080000;
        }
        public static final class layout {
            public static final int activity_main=0x7f030000;
        }
        public static final class menu {
            public static final int activity_main=0x7f070000;
        }
        public static final class string {
            public static final int app_name=0x7f050000;
            public static final int hello_world=0x7f050001;
            public static final int menu_settings=0x7f050002;
            public static final int title_activity_main=0x7f050003;
        }
        public static final class style {
            public static final int AppTheme=0x7f060000;
        }
}
```

注意：

R.java 文件是只读文件，开发人员不能对其进行修改。

在上面的代码中，程序自动为 res 目录中的资源生成了资源索引，如 res 目录中的 ic_action_search.png 文件生成了相应的图片索引，res 目录中的 activity_main.xml 文件生成了 activity_main 索引，res 目录中 strings.xml 文件中定义的 hello_world 和 app_name 等字符串资源也生成了相应的索引，那么，如何在.java 源文件中使用定义的这些资源呢？用户可以通过“R.资源类名.索引”方式来调用定义的资源。

例如，在.java 源文件中调用 main.xml 布局文件的代码如下：

```
setContentView(R.layout.activity_main);
```

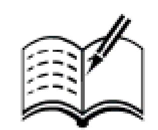
而如果要调用 hello_world 和 app_name 这两个字符串资源，则首先需要使用 Context 的 getResources()方法创建一个 Resources 对象，然后再通过该对象的 getString()方法访问其索引获取字符串值。

例如，在.java 源文件中获取 hello 和 app_name 这两个字符串的代码如下：

```
Resources resources=this.getResources();
String strhello=resources.getString(R.string.hello_world);
String strappnameString=resources.getString(R.string.app_name);
```



4.1.4　AndroidManifest.xml 文件



Note

AndroidManifest.xml 文件包含了 Androic 程序中所使用的 Activity、Service 和 Receiver 等，在该文件中，可以通过<intent-filters>标记设置默认启动的 Activity。

例如，Hello Android 程序中的 AndroidManifest.xml 文件代码如下：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.xiaoke.helloandroid.activity"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml 文件中的重要元素及说明如表 4.2 所示。

表 4.2　AndroidManifest.xml 文件中的重要元素及说明

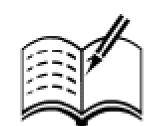
元　素	说　明
manifest	根节点，描述了 package 中所有的内容
xmlns:android	包含命名空间的声明，其属性值为 http://schemas.android.com/apk/res/android，表示 Android 中的各种标准属性能在该 xml 文件中使用，它提供了大部分元素中的数据
package	声明应用程序包
uses-sdk	应用程序所使用的 Android SDK 版本
application	包含 package 中 application 级别组件声明的根节点，一个 manifest 中可以包含零个或者一个该元素
android:icon	应用程序图标
android:label	应用程序名称
activity	与用户交互的主要工具，它是用户打开一个应用程序的初始页面
android:name	Activity 的名称
intent-filter	声明指定的一组组件支持的 Intent 值
action	组件支持的 Intent Action
category	组件支持的 Intent Category，这里通常用来指定应用程序默认启动的 Activity





注意：

在 Android 程序中，每一个 Activity 都需要在 AndroidManifest.xml 文件中有一个对应的 <activity>标记，同理，每一个 Service 也需要在 AndroidManifest.xml 文件中有一个对应的 <service>标记。



4.2　Android 程序的生命周期

Android 程序创建时，系统会自动在其.java 源文件中重写 Activity 类的 onCreate()方法，该方法是创建 Activity 时必须调用的一个方法，另外，Activity 类中还提供了如 onStart()、onResume()、onPause()、onStop()、onRestart()和 onDestroy()等方法，这些方法的先后执行顺序构成了 Android 程序的一个完整生命周期。图 4.5 是 Android 官方给出的 Android 程序生命周期图。

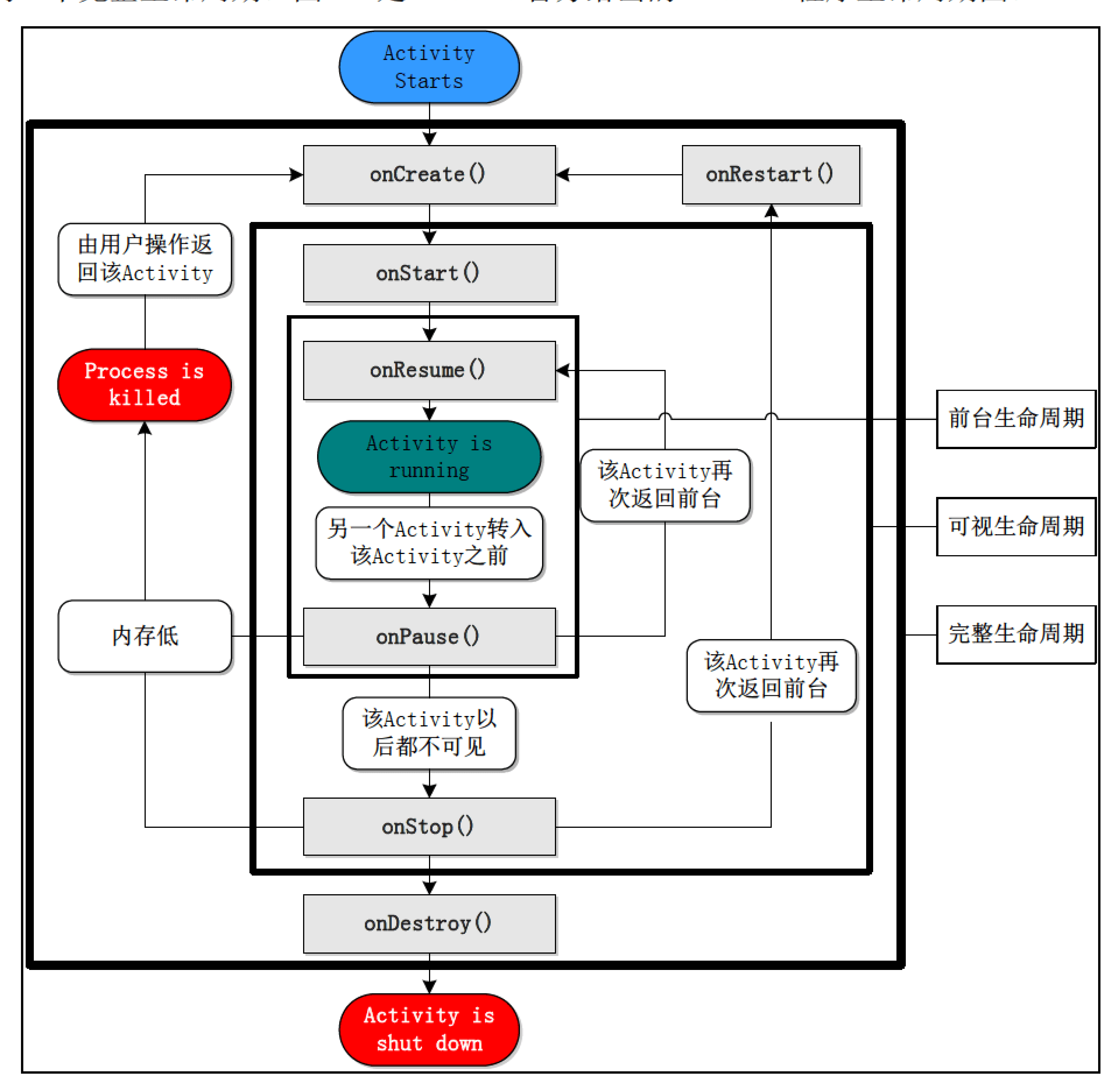
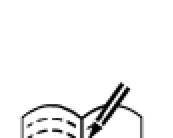


图 4.5　Android 程序的生命周期





Note

说明：

本节只需要简单了解 Android 程序的生命周期即可，本书将在第 8 章对 Activity 类及其生命周期中涉及的相关方法进行详细讲解。

4.3　Android 程序的基本组件

上面讲解了 Android 程序的目录结构及其生命周期，而要进行 Android 程序开发，还需要熟练掌握 Android 程序的基本组件。Android 程序有 4 大基本组件，即 Activity、BroadcastReceiver、Content Provider 和 Service，这里将首先对它们进行简单了解，在本书后面的章节中会分别对它们进行深入分析。

4.3.1　Activity（活动窗口）

Activity 是 Android 程序中最基本的模块，它是为用户操作而展示的可视化用户界面，一个 Android 应用程序中可以只有一个 Activity，也可以包含多个，每个 Activity 的作用及其数目，取决于应用程序及其设计。例如，可以使用一个 Activity 展示一个菜单项列表供用户选择，也可以显示一些包含说明的照片等。

在 Android 程序中，每个 Activity 都被给予一个默认的窗口以进行绘制，一般情况下，这个窗口是满屏的，但它也可以是一个小的、位于其他窗口之上的浮动窗口。

技巧：

一个 Activity 也可以使用超过一个的窗口——例如，在 Activity 运行过程中弹出的一个供用户反应的小对话框，或者，当用户选择了屏幕上特定项目后显示的必要信息。

Activity 窗口显示的可视内容是由一系列视图构成的，这些视图均继承自 View 基类。每个视图均控制着窗口中一块特定的矩形空间，父级视图包含并组织其子视图的布局，而底层视图则在它们控制的矩形中进行绘制，并对用户操作做出响应，所以，视图是 Activity 与用户进行交互的界面。例如，开发人员可以通过视图显示一个图片，然后在用户单击它时产生相应的动作。

说明：

Android 中有很多既定的视图供开发人员直接使用，如按钮、文本域、卷轴、菜单项和复选框等。

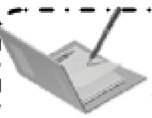
【例 4.1】 创建一个 Android 程序，首先修改 activity_main.xml 布局文件的背景图片，然后在该程序的默认 Activity 中显示 activity_main.xml 布局文件。



 实例位置：光盘\MR\Instance\04\4.1

activity_main.xml 布局文件的主要代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:background="@drawable/back"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:padding="@dimen/padding_medium"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />
</RelativeLayout>
```



说明：

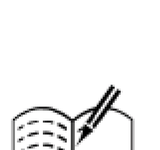
上面代码中的 android:background="@drawable/back"用来为 LinearLayout 布局设置背景。

默认 Activity 的 java 代码如下：

```
package com.xiaoke.helloandroid.activity;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.support.v4.app.NavUtils;
public class MainActivity extends Activity {
    @SuppressWarnings("null")
        @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.activity_main, menu);
        return true;
    }
}
```

运行本实例，运行效果如图 4.6 所示。




Note

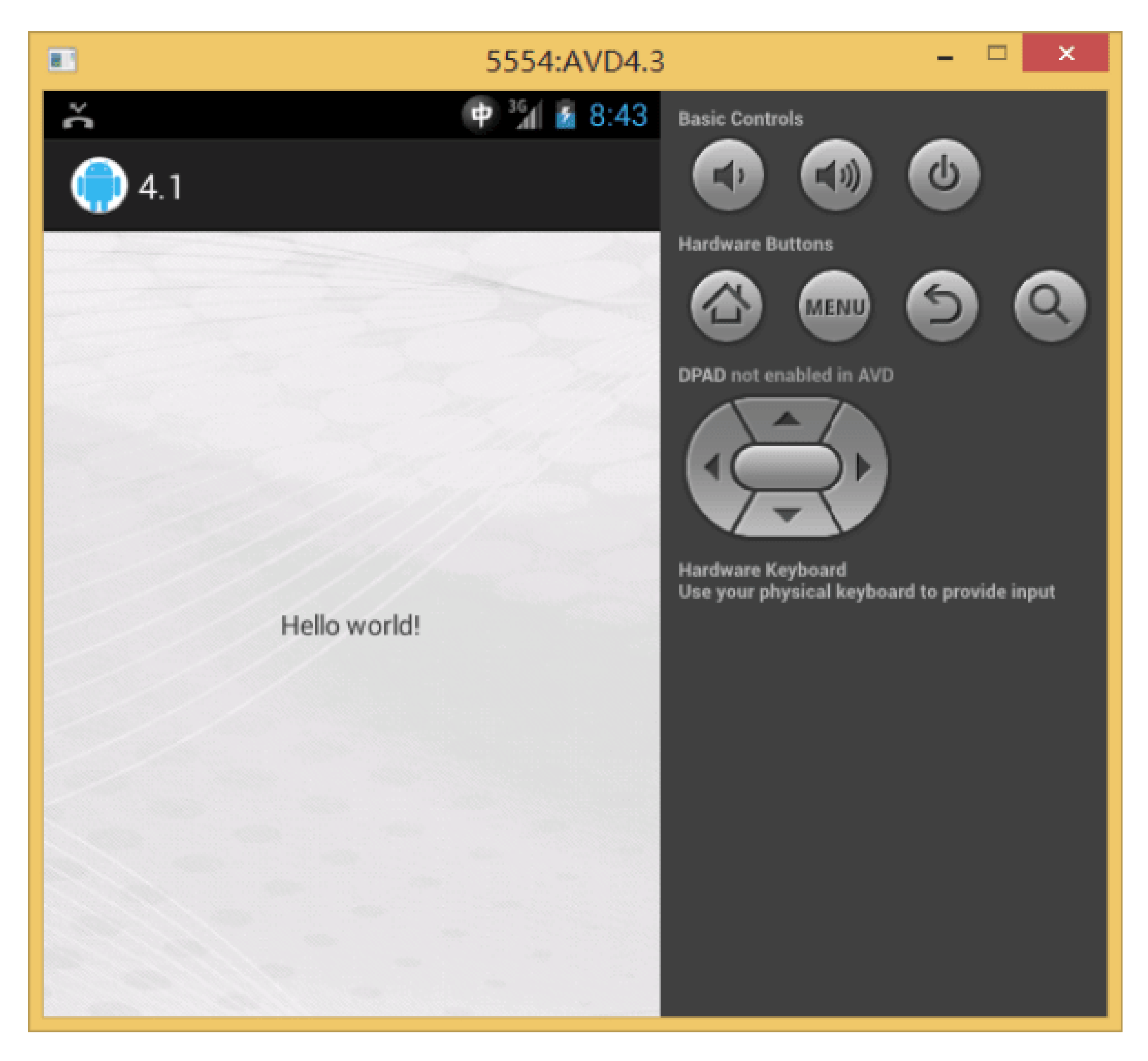


图 4.6　改变默认布局文件的背景图片并在 Activity 中显示

4.3.2　BroadcastReceiver（广播接收器）

BroadcastReceiver（广播接收器）是一个专注于接收广播通知信息，并做出对应处理的组件。Android 程序中的很多广播是源自于系统的，例如，通知时区改变、电池电量低、拍摄了一张照片或者用户改变了语言选项等；另外，Android 应用程序也可以进行广播，例如，可以在下载程序中通知其他应用程序数据下载完成等。

在一个 Android 应用程序中可以拥有任意数量的广播接收器，以对所有它感兴趣的通知信息予以响应，所有的广播接收器均继承自 BroadcastReceiver 基类。

广播接收器没有用户界面，然而它们可以启动一个 Activity 来响应收到的信息，或者用 NotificationManager 来通知用户。

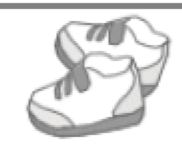
技巧：

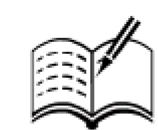
广播通知可以用很多种方式来吸引用户的注意力，如闪动背灯、震动和播放声音等。一般来说是在状态栏上放一个持久的图标，用户可以打开它并获取消息。

【例 4.2】 创建一个 Android 程序，在该程序中创建一个 BroadcastReceiverTest.java 类文件，并使该类继承自 BroadcastReceiver 类，以便作为广播接收器。BroadcastReceiverTest.java 类文件的代码如下：

 **实例位置：光盘\MR\Instance\04\4.2**

```
package com.xiaoke.exam0402.activity;
import android.content.BroadcastReceiver;
```






```
import android.content.Context;
import android.content.Intent;
public class BroadcastReceiverTest extends BroadcastReceiver {
    @Override
    public void onReceive(Context arg0, Intent arg1) {
    //TODO Auto-generated method stub
    }
}
```

4.3.3　Content Provider（数据共享）

Content Provider 是一个用来提供数据共享的组件，它主要将一些特定的应用程序数据提供给其他应用程序使用，这些应用程序数据可以存储于文件系统或者 SQLite 数据库中。在 Android 程序中，共享数据的实现需要继承自 ContentProvider 基类，该基类为其他应用程序使用和存储数据实现了一套标准方法，然而应用程序并不直接调用这些方法，而是使用一个 ContentResolver 对象，并通过调用它的方法作为替代，ContentResolver 对象提供了 query()、insert()及 update()等方法，可以对共享的数据执行各种操作。

说明：

每当出现一个需要被特定组件处理的请求时，Android 会确保那个组件的应用程序进程处于运行状态，或在必要时启动它，并确保那个相应组件的实例的存在，必要时会创建那个实例。

【例 4.3】 创建一个 Android 程序，通过使用 ContentResolver 对象的 query()方法获取 Android 系统中存储的联系人编号及姓名。

实例位置：光盘\MR\Instance\04\4.3

activity_main.xml 布局文件的主要代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/back"
    >
<TextView
    android:id="@+id/txtInfo"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#000000"
    />
</LinearLayout>
```





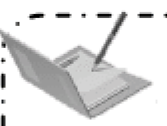
说明：

上面代码中的“android:id="@+id/txtInfo"”用来为 TextView 设置 id，而“android:textColor="#000000"”用来设置 TextView 的文本颜色。

Note

主 Activity 的 Java 代码如下：

```
public class MainActivity extends Activity {
    TextView txtView;                                          //定义 TextView 对象
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtView=(TextView) findViewById(R.id.txtInfo);         //获取布局文件中的 TextView 对象
        String strInfo="";                                     //记录联系人
        ContentResolver cResolver=getContentResolver();        //创建 ContentResolver 对象
        //使用 ContentResolver 的 query()方法获取联系人信息
        Cursor
cursor=cResolver.query(ContactsContract.Contacts.CONTENT_URI,null,null,null,null);
        while (cursor.moveToNext()) {                          //遍历获取到的联系人信息
          //获取联系人编号
          String strID=cursor.getString(cursor.getColumnIndex(PhoneLookup._ID));
          //获取联系人名称
          String
strName=cursor.getString(cursor.getColumnIndex(PhoneLookup.DISPLAY_NAME));
              strInfo+="编号："+strID+"    姓名："+strName+"\n"; //组合联系人信息
          }
          cursor.close();                                      //关闭 Cursor 对象
          txtView.setText(strInfo);                            //为 TextView 设置文本
    }
}
```



说明：

本实例中用到了 TextView 组件、ContentResolver 类和 Cursor 类，其中，TextView 组件是一个文本显示组件，ContentResolver 类用来对数据进行操作，Cursor 类类似一个集合，用来存储数据，这些内容在后面的相应章节中会进行详细讲解，这里只要简单了解其用法即可。

本程序由于需要访问系统中的联系人列表，所以需要在 AndroidManifest.xml 文件中设置联系人列表的访问权限。其代码如下：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

运行本实例，运行效果如图 4.7 所示。



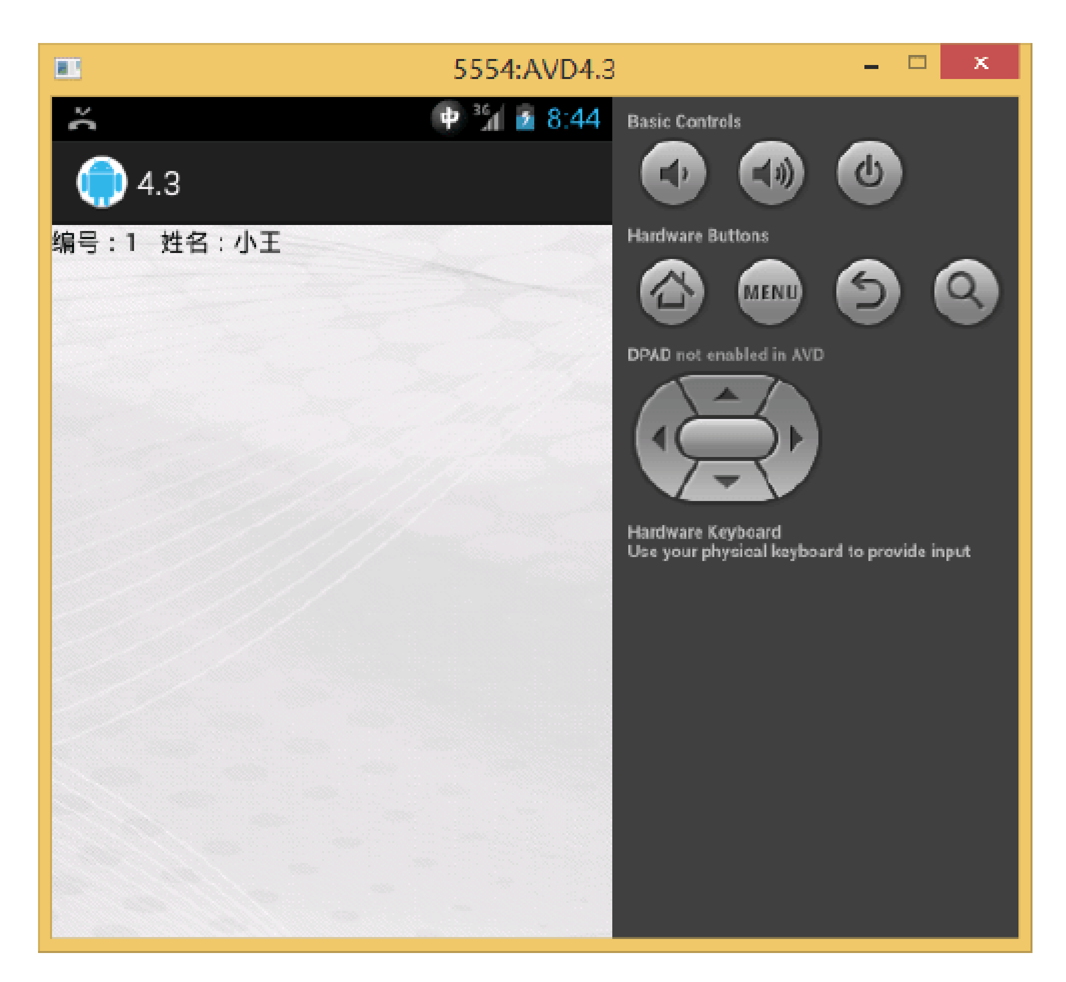


图 4.7　获取 Android 系统中的联系人并显示

4.3.4　Service（服务）

Service 是服务的意思，它没有可视化的用户界面，而是在一段时间内在后台运行。例如，一个服务可以在用户做其他事情时在后台播放背景音乐、从网络上获取一些数据或者计算一些东西并提供给需要这个运算结果的 Activity 使用。Android 程序中的每个服务都继承自 Service 基类。

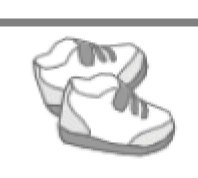
说明：

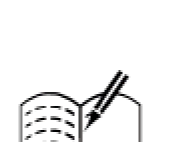
如同 Activity 和其他组件一样，服务运行于应用程序进程的主线程内。所以它不会对其他组件或用户界面有任何干扰，它们一般会派生一个新线程来进行一些耗时任务（如音乐回放等）。

例如，开发人员可以在 Android 模拟器的“设置”/“应用”/“正在运行”中查看当前正在运行的服务，但是并没有显示界面，如图 4.8 所示。



图 4.8　查看 Android 模拟器中正在运行的服务





Note

技巧：

开发媒体播放器时，经常将音乐的播放功能作为一个服务，这样当用户打开或使用其他程序时，同时还可以在后台启动音乐的播放服务，做到工作与娱乐两不误。

4.4 综 合 应 用

4.4.1 在 Android 程序中添加 Activity

【例 4.4】 本实例要求在一个 Android 程序中添加 Activity，并将新添加的 Activity 命名为 Activity02。

实例位置：光盘\MR\Instance\04\4.4

在 Android 程序中添加 Activity 时，只需要选中 Android 程序中的包文件，单击鼠标右键，在弹出的快捷菜单中选择 New/Class 命令，然后在弹出的 New Java Class 对话框中将 Superclass 设置为 android.app.Activity，并将 Name 设置为 Activity02 即可，如图 4.9 所示。

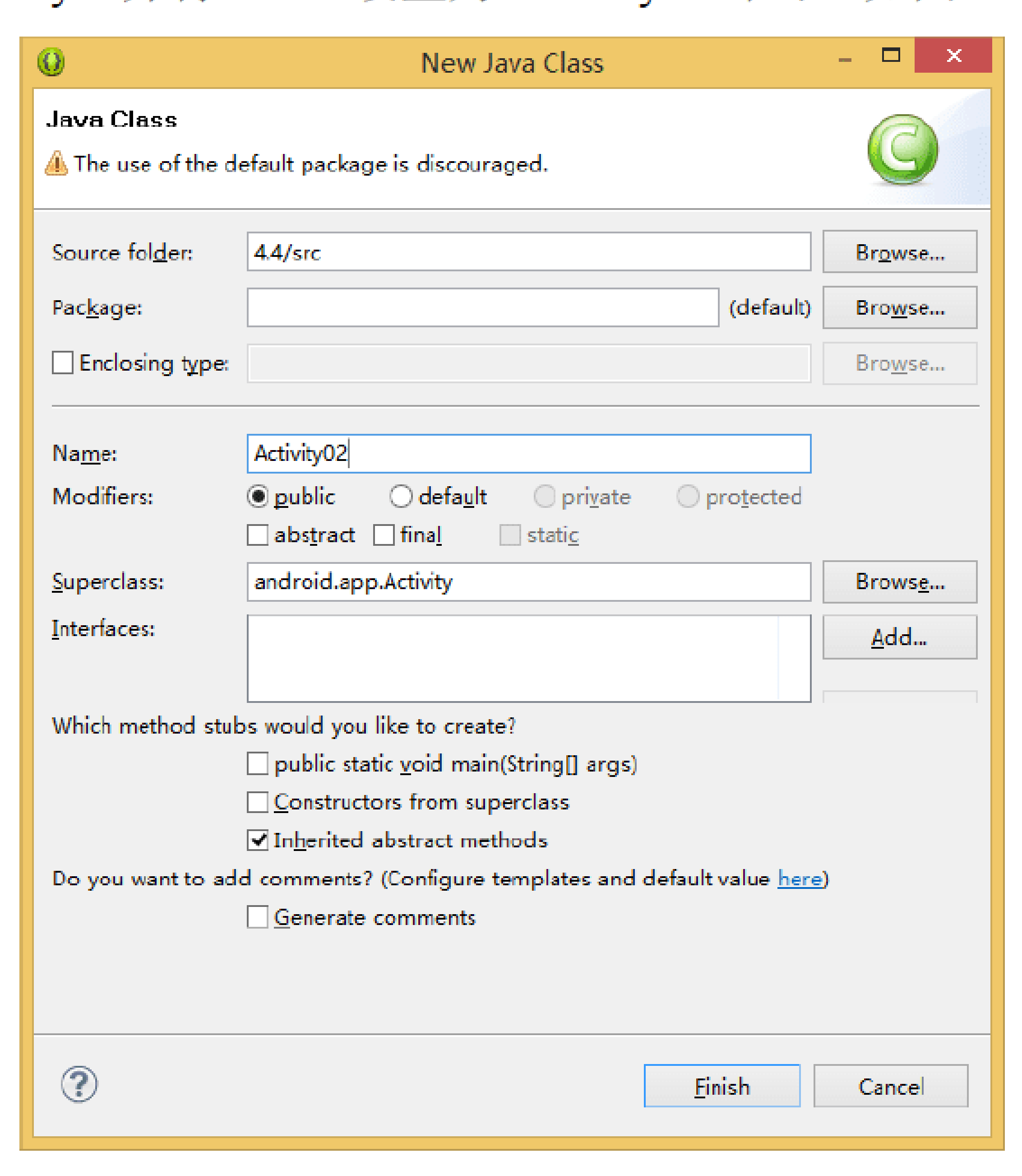


图 4.9　在 Android 程序中添加 Activity

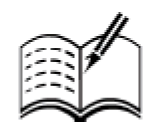
创建的 Activity 默认代码如下：

```
package com.xiaoke.sj1.activity;
import android.app.Activity;
```



```
public class Activity02 extends Activity {
}
```

4.4.2　在 Android 程序中添加 Service



【例 4.5】 本实例要求在 Android 程序中添加一个 Service 服务，并将新添加的 Service 命名为 MyService。

实例位置：光盘\MR\Instance\04\4.5

在 Android 程序中添加 Service 时，只需要选中 Android 程序中的包文件，单击鼠标右键，在弹出的快捷菜单中选择 New/Class 命令，然后在弹出的 New Java Class 对话框中将 Superclass 设置为 android.app.Service，并将 Name 设置为 MyService 即可，如图 4.10 所示。

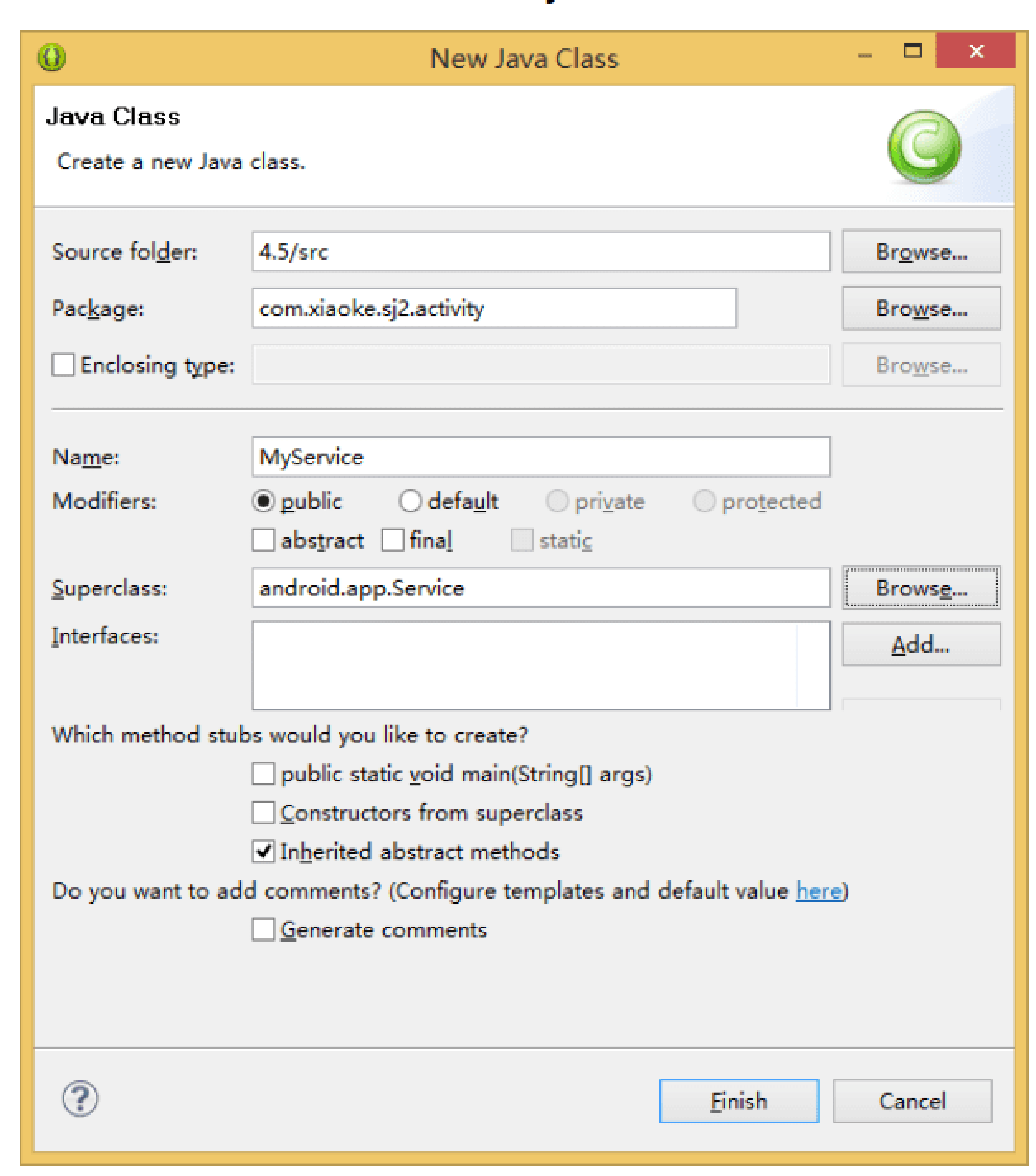
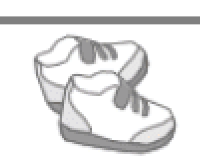


图 4.10　在 Android 程序中添加 Service

创建的 Service 默认代码如下：

```
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
public class MyService extends Service {
	@Override
	public IBinder onBind(Intent arg0) {
		//TODO Auto-generated method stub
```



```
            return null;
        }
}
```

Note

4.5　本章常见错误

运行 Android 程序时，发现无法加载 XML 布局文件，并提示 java.lang.NullPointerException 的异常，这主要是由于忘记加载 Activity 的 layout 文件造成的，解决该错误，只需要在.java 文件中添加如下代码即可。

```
setContentView(R.layout.layout 布局文件名称);
```

4.6　本 章 小 结

本章重点讲解了一个 Android 程序的完整目录结构，然后简单介绍了 Android 程序的生命周期及其基本的 4 大组件，学习本章内容时，读者需要重点掌握 Android 程序的目录结构，并掌握各个目录及文件的使用；而对于 Android 程序的生命周期及其 4 大组件，这里只需要简单了解即可，后面会有专门的章节对它们进行详细讲解。

4.7　跟 我 上 机

参考答案：光盘\MR\跟我上机

创建一个 Android 程序，命名为 04，默认包名为 com.mingrisoft，然后在该程序的 src 目录下再创建一个 com.mingrisoft02 的包。其具体步骤为：选中 src 目录，单击鼠标右键，在弹出的快捷菜单中选择 New/Package 命令，在弹出的 New Java Package 对话框的 Name 文本框中输入“com.mingrisoft02”，然后单击 Finish 按钮即可。

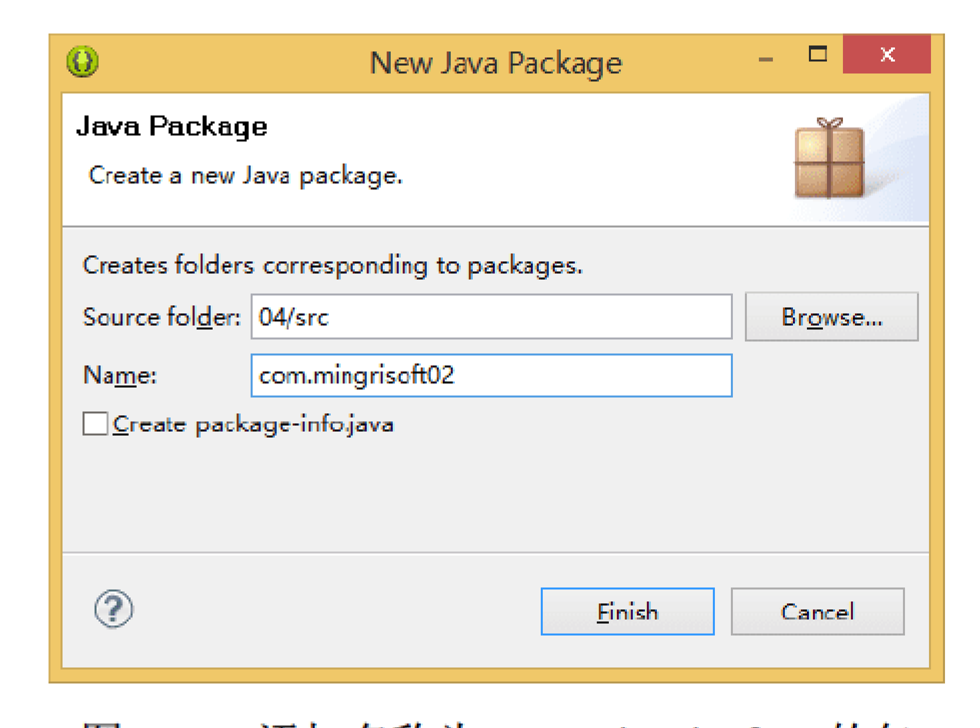


图 4.11　添加名称为 com.mingrisoft02 的包



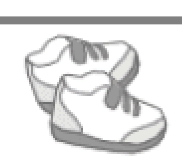
Android 常用组件的使用

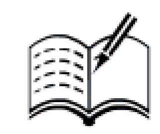
（视频讲解：2 小时 42 分钟）

组件是 Android 程序设计的基本组成单位，通过使用组件可以高效地开发 Android 应用程序，所以，熟练掌握组件的使用是合理、有效地进行 Android 程序开发的重要前提。本章将首先对 Android 程序的 UI 界面进行介绍，然后对 Android 应用程序开发中的常用组件进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- □ 使用 XML 布局文件实现游戏的开始界面
- □ 通过 Java 代码实现游戏的进入界面
- □ 在窗体中垂直显示 4 张图片
- □ 应用 TextView 显示多种样式的文本
- □ 使用 EditText 组件实现用户注册信息的输入
- □ 使用 AutoCompleteTextView 组件实现自动提示功能
- □ 添加选择性别的单选按钮
- □ 使用适配器为 ListView 设置列表项
- □ 显示列表选择框并获取其选择项
- □ 使用 ImageView 显示图像
- □ 使用 ImageSwitcher 组件实现简单的图片查看器
- □ 实现带图标的 ListView 列表
- □ 猜猜鸡蛋放在哪只鞋子里





Note

5.1 Android 的 UI 界面

用户界面设计（UI）是 Android 应用开发中最基本，也是最重要的内容，在设计用户界面时，首先需要了解界面中的 UI 元素如何呈现给用户，也就是如何控制 UI 界面。在 Android 中，提供了 4 种控制 UI 界面的方法，下面分别进行介绍。

5.1.1 Android UI 界面概述

在 Android 中，所有的 UI 界面都是由 View 和 ViewGroup 类及其子类组合而成的。其中，View 类是所有 UI 组件的基类，它是 Android 平台中用户界面体现的基础单位，它提供了如文本输入框和按钮之类的 UI 对象的完整实现；而 ViewGroup 类是容纳这些 UI 组件的容器，它提供了像流式布局、表格布局以及相对布局之类的布局架构，而且 ViewGroup 类本身也是 View 类的子类。在 ViewGroup 类中，除了可以包含普通的 View 类外，还可以再次包含 ViewGroup 类。View 和 ViewGroup 类的层次结构如图 5.1 所示。

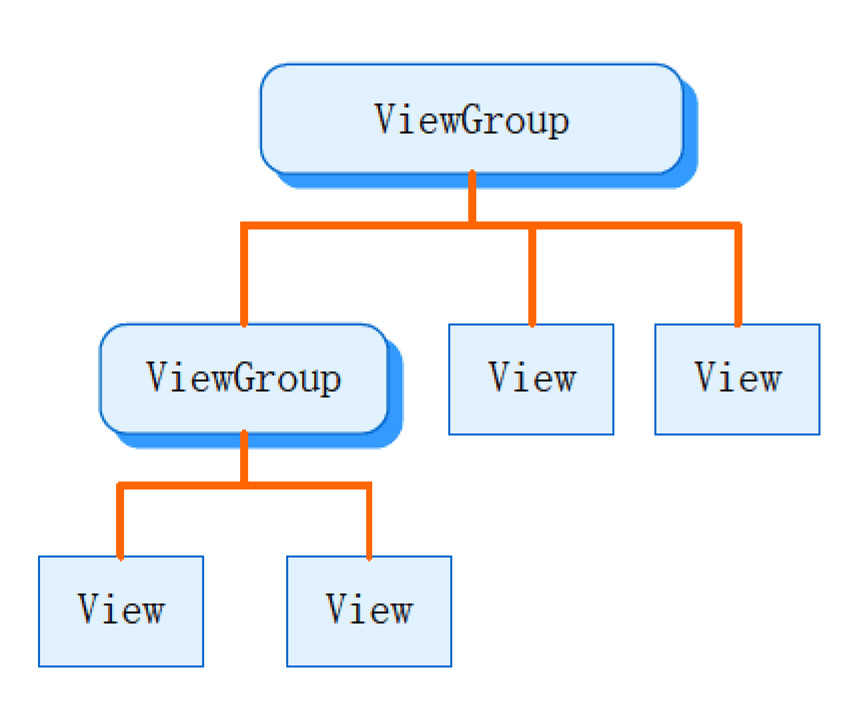
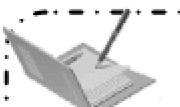


图 5.1　Android UI 界面的层次结构



说明：

View 类是一个数据体，它的属性存储了用于屏幕上一块矩形区域的布局参数及内容，并负责它所辖的这个矩形区域之中所有测量、布局、焦点转换、卷动以及按键/触摸手势的处理。作为一个用户界面对象，View 同时也担任着用户交互关键点以及交互事件接受者的角色。

5.1.2 使用 XML 布局文件控制 UI 界面

在 Android 中，提供了一种非常简单、方便的方法用于控制 UI 界面，该方法采用 XML 文件来进行界面布局，从而将布局界面的代码和逻辑控制的 Java 代码分离开来，使得程序的结构更加清晰、明了。

使用 XML 布局文件控制 UI 界面可以分为以下两个关键步骤。

（1）在 Android 程序的 res\layout 目录下编写 XML 布局文件，XML 布局文件名可以是任何符合 Java 命名规则的文件名。创建后，R.java 会自动收录该布局资源。

（2）在 Activity 中使用以下 Java 代码显示 XML 文件中布局的内容。

```
setContentView(R.layout.main);
```



在上面的代码中，main 是 XML 布局文件的文件名。

通过上面的代码步骤即可轻松实现 UI 界面的布局并显示功能。下面通过一个具体的实例来演示如何使用 XML 布局文件来控制 Android 程序的 UI 界面。

【例 5.1】 在 Eclipse 中创建 Android 项目，使用 XML 布局文件实现游戏的开始界面。



实例位置：光盘\MR\Instance\05\5.1

首先修改 Android 项目的 res\layout 目录下的布局文件 main.xml，在该文件中，采用帧布局（FrameLayout），并且添加两个 TextView 组件，第一个用于显示提示文字，第二个用于在窗体的正中间位置显示开始游戏按钮。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/back"
    >
     <!-- 添加提示文字 -->
     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/title"
         android:textSize="20dp"
         android:textColor="#111111"
    />
    <!-- 添加开始按钮 -->
    <TextView
         android:id="@+id/startButton"
         android:layout_gravity="center_vertical|center_horizontal"
         android:text="@string/start"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:textSize="20dp"
         android:textColor="#111111"
    />
</FrameLayout>
```

说明：

上面的代码中用到了帧布局（FrameLayout），该布局将会在第 5 章进行详细讲解，使用该布局方式时，可以使用 “android:layout_gravity="center_vertical|center_horizontal"” 使指定组件在帧布局中居中显示。

然后，修改 res\values 目录下的 strings.xml 文件，并且在该文件中添加两个字符串常量，分别用来表示标题内容和开始按钮的内容。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <string name="hello">Hello World, 5.1Activity!</string>
    <string name="app_name">5.1</string>
    <string name="title">使用 XML 布局文件控制 UI 界面</string>
    <string name="start">单击开始游戏......</string>
</resources>
```


Note



说明：

strings.xml 文件用于定义程序中应用的字符串常量，其中，每一个<string>子元素都可以定义一个字符串常量，常量名称由 name 属性指定，常量内容写在起始标记<string>和结束标记</string>之间。

最后还需要在主活动中，也就是主 Activity 中，使用以下代码指定活动应用的布局文件。

```
setContentView(R.layout.main);
```



说明：

在应用 Eclipse 创建 Android 项目时，Eclipse 会自动在主活动的 onCreate()方法中添加指定布局文件 main.xml 的代码。

运行本实例，将显示如图 5.2 所示的运行效果。



图 5.2　使用 XML 文件布局游戏开始界面

5.1.3　在 Java 代码中控制 UI 界面

在 Android 中，支持像 Java Swing 那样完全通过代码控制 UI 界面，也就是所有的 UI 组件都通过 new 关键字创建出来，然后将这些 UI 组件添加到布局管理器中，从而实现用户界面。

在代码中控制 UI 界面可以分为以下 3 个关键步骤。

（1）创建布局管理器，可以是帧布局、表格布局、线性布局和相对布局等，并且设置布局管理器的属性，如为布局管理器设置背景图片等。

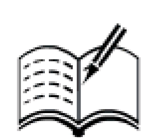
（2）创建具体的组件，可以是 TextView、ImageView、EditText 和 Button 等任何 Android 提供的组件，并且设置组件的布局和各种属性。



（3）将创建的具体组件添加到布局管理器中。

下面通过一个具体的实例来演示如何使用 Java 代码控制 Android 的 UI 界面。

【例 5.2】 在 Eclipse 中创建 Android 项目，完全通过 Java 代码实现游戏的进入界面。



实例位置：光盘\MR\Instance\05\5.2

程序的开发步骤如下：

（1）在新创建的 Android 项目中，打开\src\com\xiaoke\exam0502\activity 目录下的 MainActivity.java 文件，然后将默认生成的下面这行代码删除。

```
setContentView(R.layout.main);
```

（2）在 MainActivity 的 onCreate()方法中，创建一个帧布局管理器，并为该布局管理器设置背景。其关键代码如下：

```
FrameLayout frameLayout = new FrameLayout(this);          //创建帧部局管理器
frameLayout.setBackgroundDrawable(this.getResources().getDrawable(
            R.drawable.back));                             //设置背景
setContentView(frameLayout);                               //设置在 Activity 中显示 frameLayout
```

（3）创建一个 TextView 组件 text1，设置其文字大小和颜色，并将其添加到布局管理器中。其具体代码如下：

```
TextView text1 = new TextView(this);
text1.setText("在 Java 代码中控制 UI 界面");                //设置显示的文字
text1.setTextSize(TypedValue.COMPLEX_UNIT_PX, 20);        //设置文字大小，单位为像素
text1.setTextColor(Color.rgb(1, 1, 1));                    //设置文字的颜色
frameLayout.addView(text1);                                //将 text1 添加到布局管理器中
```

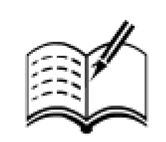
（4）创建一个 TextView 组件 text2，设置其显示文字、文字大小、颜色和布局，并将其添加到布局管理器中。其具体代码如下：

```
TextView text2 = new TextView(this);
text2.setText("单击进入游戏......");                        //设置显示文字
text2.setTextSize(TypedValue.COMPLEX_UNIT_PX, 20);        //设置文字大小，单位为像素
text2.setTextColor(Color.rgb(1, 1, 1));                    //设置文字的颜色
LayoutParams params = new LayoutParams(
      ViewGroup.LayoutParams.WRAP_CONTENT,
      ViewGroup.LayoutParams.WRAP_CONTENT);                //创建保存布局参数的对象
params.gravity = Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL;    //设置居中显示
text2.setLayoutParams(params);                             //设置布局参数
frameLayout.addView(text2);                                //将 text2 添加到布局管理器中
```

说明：

在通过 setTextSize()方法设置 TextView 的文字大小时，可以指定使用的单位，在上面的代码中，int 型的常量 TypedValue.COMPLEX_UNIT_PX 表示单位是像素，如果要设置单位是磅，可以使用常量 TypedValue.COMPLEX_UNIT_PT，这些常量可以在 Android 官方提供的 API 中找到。



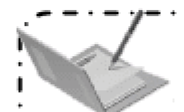


Note

运行本实例，将显示如图 5.3 所示的运行效果。



图 5.3　通过 Java 代码布局游戏开始界面



说明：

完全通过 Java 代码控制 UI 界面，虽然该方法比较灵活，但是其开发过程比较烦琐，而且不利于高层次的解耦，因此不推荐采用这种方式控制 UI 界面。

5.1.4　使用 XML 和 Java 代码混合控制 UI 界面

前面两节中介绍了完全通过 XML 布局文件控制 UI 界面和完全通过 Java 代码控制 UI 界面，虽然通过第一种方法实现比较方便、快捷，但是该方法有失灵活，而第二种方法虽然比较灵活，但是开发过程比较烦琐，鉴于这两种方法的优缺点，我们来看另一种控制 UI 界面的方法，那就是使用 XML 和 Java 代码混合控制 UI 界面。

使用 XML 和 Java 代码混合控制 UI 界面，习惯上把变化小、行为比较固定的组件放在 XML 布局文件中，把变化较多、行为控制比较复杂的组件交给 Java 代码来管理。下面就通过一个具体的实例来演示如何使用 XML 和 Java 代码混合控制 UI 界面。

【例 5.3】 在 Eclipse 中创建 Android 项目，通过 XML 和 Java 代码在 Android 程序窗体中垂直显示 4 张图片。

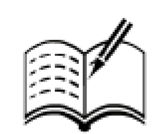
实例位置：光盘\MR\Instance\05\5.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认创建的<TextView>组件删除，然后将默认创建的线性布局的 orientation 属性值设置为 vertical（垂直），并且为该线性布局设置背景以及 id 属性。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
```





```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/back"
        android:id="@+id/layout"
        >
</LinearLayout>
```

（2）在 MainActivity 中，声明 img 和 imagePath 两个属性，其中，img 是一个 ImageView 类型的一维数组，用于保存 ImageView 组件；imagePath 是一个 int 型的一维数组，用于保存要访问的图片资源。其关键代码如下：

```
private  ImageView[] img=new ImageView[4];            //声明一个保存 ImageView 组件的数组
private int[] imagePath=new int[]{
R.drawable.img01,R.drawable.img02,R.drawable.img03,R.drawable.img04
};                                                    //声明并初始化一个保存访问图片的数组
```

（3）在 MainActivity 的 onCreate()方法中，首先获取在 XML 布局文件中创建的线性布局管理器，然后通过一个 for 循环创建 4 个显示图片的 ImageView 组件，并将其添加到布局管理器中。其关键代码如下：

```
setContentView(R.layout.main);
//获取 XML 文件中定义的线性布局管理器
LinearLayout layout=(LinearLayout)findViewById(R.id.layout);
for(int i=0;i<imagePath.length;i++){
    img[i]=new ImageView(this);                         //创建一个 ImageView 组件
    img[i].setImageResource(imagePath[i]);              //为 ImageView 组件指定要显示的图片
    img[i].setPadding(5, 5, 5, 5);                      //设置 ImageView 组件的内边距
    LayoutParams params=new LayoutParams(200,120);      //设置图片的宽度和高度
    img[i].setLayoutParams(params);                     //为 ImageView 组件设置布局参数
    layout.addView(img[i]);                             //将 ImageView 组件添加到布局管理器中
}
```

运行本实例，将显示如图 5.4 所示的运行效果。

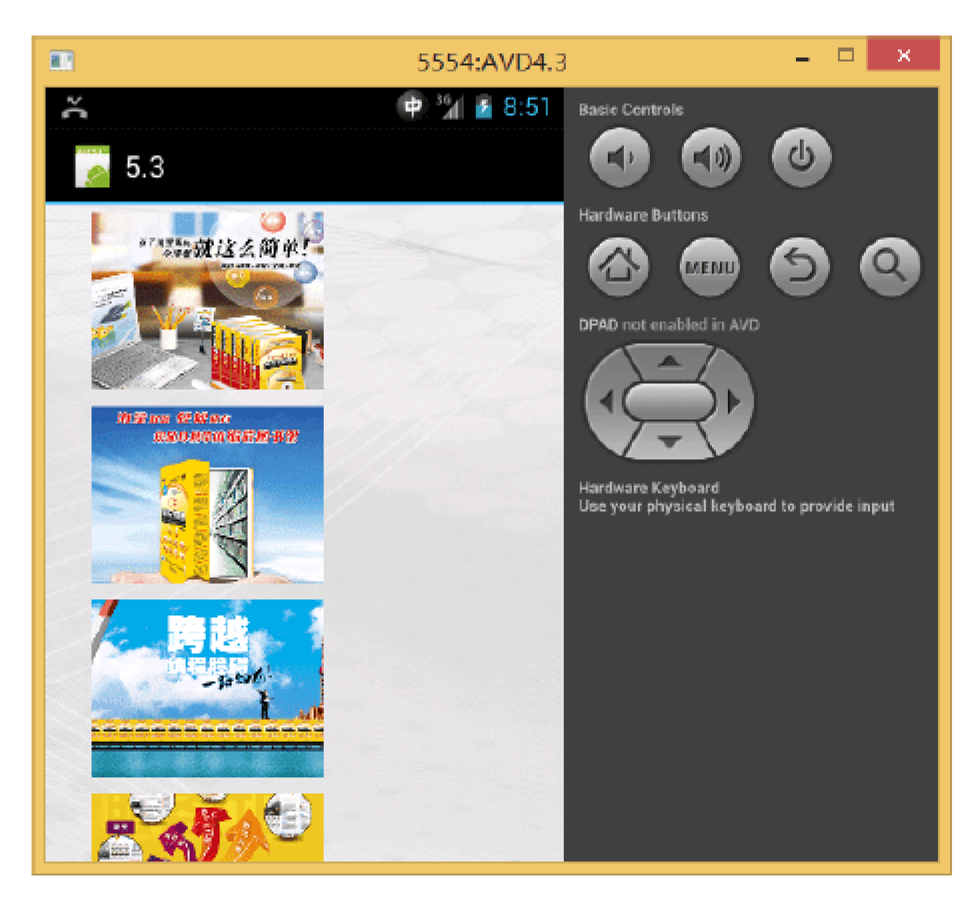
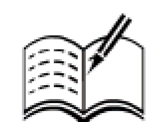


图 5.4 在窗体中垂直显示 4 张图片





5.1.5 开发自定义的 View

一般情况下，开发 Android 应用程序的 UI 界面，都不直接使用 View 和 ViewGroup 类，而是使用这两个类的子类。例如，要显示一个图片，就可以使用 View 类的子类 ImageView。虽然 Android 提供了很多继承了 View 类的 UI 组件，但是在实际开发时，还会出现不足以满足程序需要的情况。这时，就可以通过继承 View 类来开发自己的组件。开发自定义的 View 组件大致分为以下 3 个步骤。

（1）创建一个继承 android.view.View 类的 View 类，并且重写构造方法。

（2）根据需要重写相应的方法。被重写的方法可以通过下面的方法找到。

在代码中单击鼠标右键，在弹出的快捷菜单中选择“源代码”/“覆盖/实现方法”命令，将打开如图 5.5 所示的对话框，在该对话框的列表中显示出了可以被重写的方法。只需要选中要重写方法前面的复选框，并单击“确定”按钮，Eclipse 将自动重写指定的方法。通常情况下，不需要重写全部的方法。

（3）在项目的活动中，创建并实例化自定义 View 类，并将其添加到布局管理器中即可。

下面通过一个具体的实例，来演示如何开发自定义的 View。

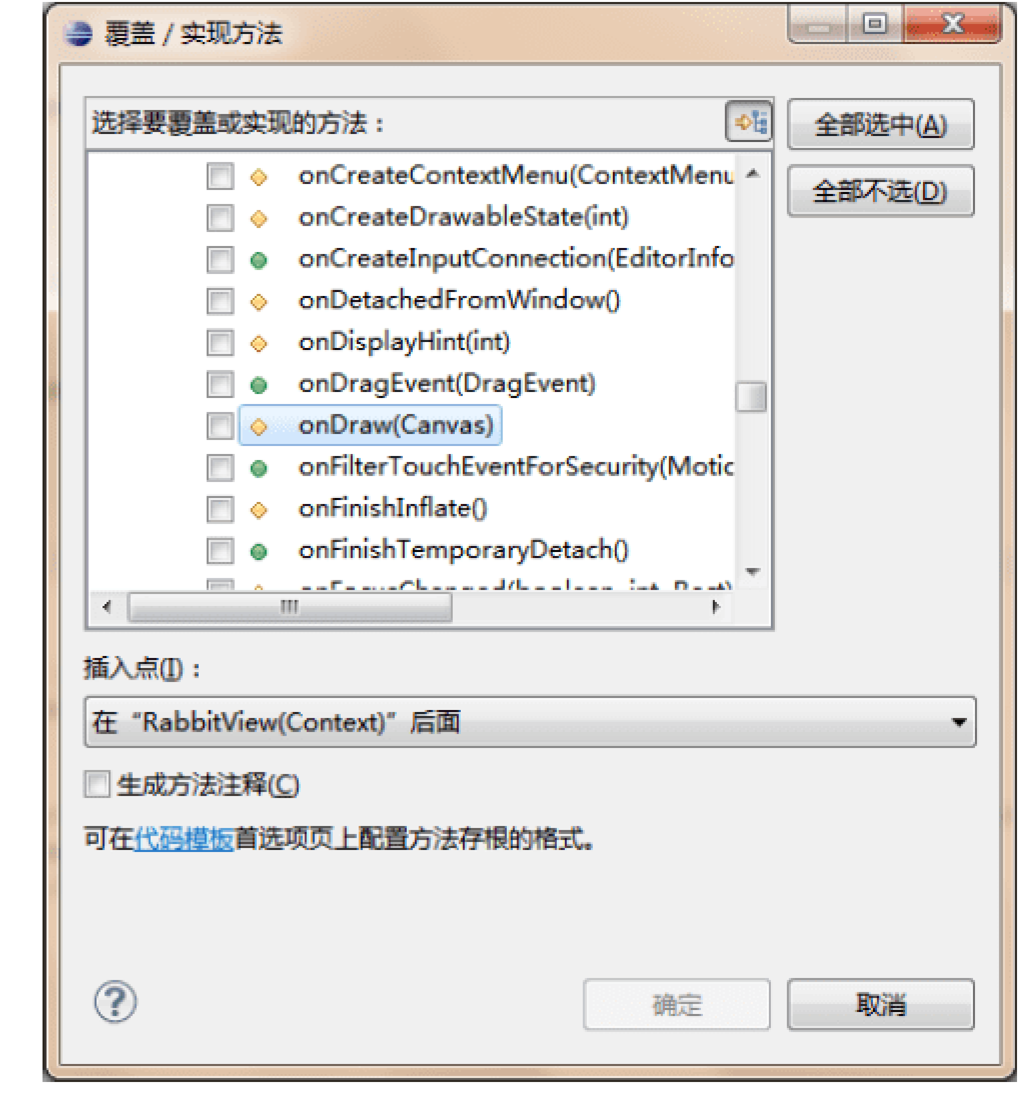


图 5.5 “覆盖/实现方法”对话框

【例 5.4】 在 Eclipse 中创建 Android 项目，通过自定义 View 组件实现 Activity 界面的切换。

实例位置：光盘\MR\Instance\05\5.4

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认创建的 LinearLayout 和 TextView 组件删除，然后添加一个帧布局组件 FrameLayout，并且设置其背景和 id 属性。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/back"
    android:id="@+id/mylayout"
    >
</FrameLayout>
```

（2）创建一个名称为 UserView 的 Java 类，该类继承自 android.view.View 类，重写带一个参数 Context 的构造方法和 onDraw()方法。其中，在 onDraw()方法中重新绘制 Activity 窗口的背景。UserView 类的关键代码如下：





```
public class UserView extends View {
    public UserView(Context context) {
        super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        Paint paint =new Paint();                                          //创建 Paint 的对象
        Bitmap bitmap = BitmapFactory.decodeResource(this.getResources(),
                R.drawable.viewback);                                      //根据图片生成位图对象
        canvas.drawBitmap(bitmap, 0, 0, paint);                            //绘制图片
        if (bitmap.isRecycled()) {                                         //判断图片是否回收
            bitmap.recycle();                                              //强制回收图片
        }
    }
}
```

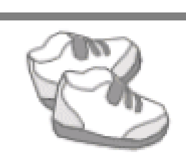
（3）在主活动的 onCreate()方法中，首先获取帧布局管理器，并创建 UserView 对象，然后为 UserView 对象添加触摸事件监听，在触摸事件中重绘 View 组件，最后将 UserView 对象添加到布局管理器中。其关键代码如下：

```
FrameLayout frameLayout=(FrameLayout)findViewById(R.id.mylayout);   //获取帧布局管理器
final UserView view=new UserView(MainActivity.this);                 //创建并实例化 RabbitView 类
//添加触摸事件监听
view.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        view.invalidate();                                           //重绘 View 组件
        return true;
    }
});
frameLayout.addView(view);                                           //将 View 添加到布局管理器中
```

运行本实例，将显示如图 5.6 所示的运行效果。



图 5.6　通过继承 View 自定义 Activity 界面





5.2 文本类组件

Android SDK 中提供了多种用于开发 Android 应用程序的组件，其中，开发人员最常用到的组件分类如表 5.1 所示。

表 5.1 常用组件的分类

组 件 分 类	作 用
文本类组件	显示文本
按钮类组件	执行操作
选择类组件	使用户可以进行单选或多选等操作
列表类组件	显示列表信息
图像类组件	以各种方式显示图像

5.2.1 TextView 组件

在 Android 中，文本框使用 TextView 表示，用于在屏幕上显示文本，这与 Java 中的文本框组件不同，它相当于 Java 中的标签，也就是 JLable。需要说明的是，Android 中的文本框组件可以显示单行文本，也可以显示多行文本，而且还可以显示带图像的文本。

在 Android 中，可以使用两种方法向屏幕中添加文本框，一种是通过在 XML 布局文件中使用<TextView>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<TextView>标记在 XML 布局文件中添加。在 XML 布局文件中添加文本框的基本语法格式如下：

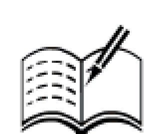
```
<TextView
属性列表
/>
```

TextView 支持的常用 XML 属性如表 5.2 所示。

表 5.2 TextView 支持的 XML 属性

XML 属性	描 述
android:autoLink	用于指定是否将指定格式的文本转换为可单击的超链接形式，其属性值有 none、web、email、phone、map 或 all
android:drawableBottom	用于在文本框内文本的底端绘制指定图像，该图像可以是放在 res\drawable 目录下的图片，通过“@drawable/文件名（不包括文件的扩展名）”设置
android:drawableLeft	用于在文本框内文本的左侧绘制指定图像，该图像可以是放在 res\drawable 目录下的图片，通过“@drawable/文件名（不包括文件的扩展名）”设置
android:drawableRight	用于在文本框内文本的右侧绘制指定图像，该图像可以是放在 res\drawable 目录下的图片，通过“@drawable/文件名（不包括文件的扩展名）”设置





续表

XML 属性	描　述
android:drawableTop	用于在文本框内文本的顶端绘制指定图像，该图像可以是放在 res\drawable 目录下的图片，通过"@drawable/文件名（不包括文件的扩展名）"设置
android:gravity	用于设置文本框内文本的对齐方式，可选值有 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal 等。这些属性值也可以同时指定，各属性值之间用竖线隔开。例如要指定组件靠右下角对齐，可以使用属性值 right|bottom
android:hint	用于设置当文本框中文本内容为空时，默认显示的提示文本
android:inputType	用于指定当前文本框显示内容的文本类型，其可选值有 textPassword、textEmailAddress、phone 和 date 等，可以同时指定多个，使用"|"进行分隔
android:singleLine	用于指定该文本框是否为单行模式，其属性值为 true 或 false，为 true 表示该文本框不会换行，当文本框中的文本超过一行时，其超出的部分将被省略，同时在结尾处添加"…"
android:text	用于指定该文本中显示的文本内容，可以直接在该属性值中指定，也可以通过在 strings.xml 文件中定义文本常量的方式指定
android:textColor	用于设置文本框内文本的颜色，其属性值可以是#rgb、#argb、#rrggbb 或#aarrggbb 格式指定的颜色值
android:textSize	用于设置文本内文本的字体大小，其属性为代表大小的数值加上单位组成，其单位可以是 px、pt、sp 和 in 等
android:width	用于指定文本的宽度，以像素为单位
android:height	用于指定文本的高度，以像素为单位



说明：

在表 5.2 中，只给出了 TextView 组件常用的部分属性，关于该组件的其他属性，可以参阅 Android 官方提供的 API 文档。

下面将给出一个关于文本框的实例。

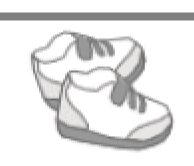
【例 5.5】 在 Eclipse 中创建 Android 项目，主要实现为文本中的 E-mail 地址添加超链接、显示带图像的文本、显不同颜色的单行文本和多行文本。

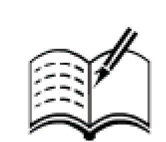
实例位置：光盘\MR\Instance\05\5.5

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，首先添加两个 TextView 组件，分别用来显示单行文本和多行文本。其代码如下：

```
<TextView
    android:id="@+id/tv1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:singleLine="true"
    android:text="单行文本：在 Android 中，文本框使用 TextView 表示，用于在屏幕上显示文本"
    android:textSize="14sp"
    android:textColor="#00FF00"
/>
```





Note

```
<TextView
    android:id="@+id/tv2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="多行文本：在 Android 中，文本框使用 TextView 表示，用于在屏幕上显示文本"
    android:textSize="14sp"
    android:textColor="#00FF00"
/>
```

（2）在 main.xml 布局文件中添加一个 TextView 组件，通过设置其 autoLink 属性使其文本显示为 E-mail 格式。其代码如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="mingrisoft@mingrisoft.com"
    android:autoLink="email"
    android:height="50px"
/>
```

（3）在 main.xml 布局文件中添加一个 TextView 组件，通过设置其 drawableTop 属性为其添加一个图片。其代码如下：

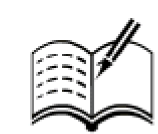
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="带图片的 TextView"
    android:drawableTop="@drawable/icon"
    android:height="50px"
/>
```

（4）在 main.xml 布局文件中添加两个 TextView 组件，这两个 TextView 组件的文本显示样式将在 java 文件中进行设置。其代码如下：

```
<TextView
    android:id="@+id/tv3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="16sp"
/>
<TextView
    android:id="@+id/tv4"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    />
```

（5）打开 MainActivity.java 文件，在 OnCreate()方法中，首先使用 findViewById()方法找到 id 为 tv3 的组件，然后使用 setText()方法为该组件设置文本，在设置文本时，用到了 HTML 标记。





其代码如下：

```
TextView tView1=(TextView) findViewById(R.id.tv3);                    //找到 tv3 组件
//使用 HTML 标记设置文本
tView1.setText(Html.fromHtml("大家好<font color=red>改变局部颜色</font>！"));
```

（6）在 MainActivity.java 文件的 OnCreate()方法中，使用 findViewById()方法找到 id 为 tv4 的组件，并使用 SpannableStringBuilder 对象的 setSpan()方法设置文本的显示样式，最后将要显示的文本通过 setText()方法显示在 id 为 tv4 的 TextView 组件中。其代码如下：

```
String str="根据段落改变文本颜色！";                                    //定义要显示的字符串
TextView tView2=(TextView) findViewById(R.id.tv4);                    //找到 tv4 组件
//创建 SpannableStringBuilder 对象
SpannableStringBuilder styleBuilder=new SpannableStringBuilder(str);
//使用 SpannableStringBuilder 对象的 setSpan()方法设置样式
styleBuilder.setSpan(new ForegroundColorSpan(Color.GREEN), 0, 4, Spannable.SPAN_EXCLUSIVE_
EXCLUSIVE);
styleBuilder.setSpan(new ForegroundColorSpan(Color.YELLOW), 4,6, Spannable.SPAN_EXCLUSIVE_
EXCLUSIVE);
styleBuilder.setSpan(new ForegroundColorSpan(Color.RED), 6, 11, Spannable.SPAN_EXCLUSIVE_
EXCLUSIVE);
tView2.setText(styleBuilder);                                         //为 tv4 组件设置文本
```

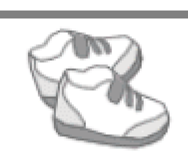
运行本实例，将显示如图 5.7 所示的运行效果。

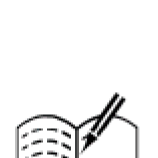


图 5.7　应用 TextView 显示多种样式的文本

5.2.2　EditText 组件

在 Android 中，编辑框使用 EditText 表示，用于在屏幕上显示文本输入框，这与 Java 中的编辑框组件功能类似。需要说明的是，Android 中的编辑框组件可以输入单行文本，也可以输入多行文本，而且还可以输入指定格式的文本（如密码、电话号码和 E-mail 地址等）。





Note

在 Android 中，可以使用两种方法向屏幕中添加编辑框：一种是通过在 XML 布局文件中使用<EditText>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<EditText>标记在 XML 布局文件中添加。在 XML 布局文件中添加编辑框的基本语法格式如下：

```
<EditText
属性列表
/>
```

由于 EditText 类是 TextView 的子类，所以对于表 5.2 中列出的 XML 属性，同样适用于 EditText 组件。特别需要注意的是，android:inputType 属性，在 EditText 组件中，通过指定该属性可以帮助输入法显示合适的类型。例如，要添加一个只能显示电话号码的编辑器，可以将 android:inputType 属性设置为 phone。下面将给出一个关于编辑框的实例。

【例 5.6】 在 Eclipse 中创建 Android 项目，主要使用 EditText 组件实现用户注册信息的输入功能。

实例位置：光盘\MR\Instance\05\5.6

修改新建项目的 res\layout 目录下的布局文件 main.xml，为默认添加的垂直线性布局管理器 LinearLayout 设置背景，并为默认添加的 TextView 组件设置高度和对其中的 E-mail 格式的文本设置超级链接。修改后的代码如下：

```
<TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入用户名："
        android:textSize="12sp"
        />
    <EditText
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:inputType="text"
        android:hint="请输入用户名"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入用户密码："
        android:textSize="12sp"
        />
    <EditText
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:inputType="textPassword"
        android:hint="请输入用户密码"
        />
    <TextView
        android:layout_width="fill_parent"
```





```
    android:layout_height="wrap_content"
    android:text="请输入出生日期："
    android:textSize="12sp"
    />
<EditText
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:inputType="date"
    android:hint="请输入出生日期"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="请输入手机号码："
    android:textSize="12sp"
    />
<EditText
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    android:maxLength="11"
    android:inputType="phone"
    android:hint="请输入手机号码"
    android:drawableLeft="@drawable/icon"
    />
```

运行本实例，将显示如图 5.8 所示的运行效果。

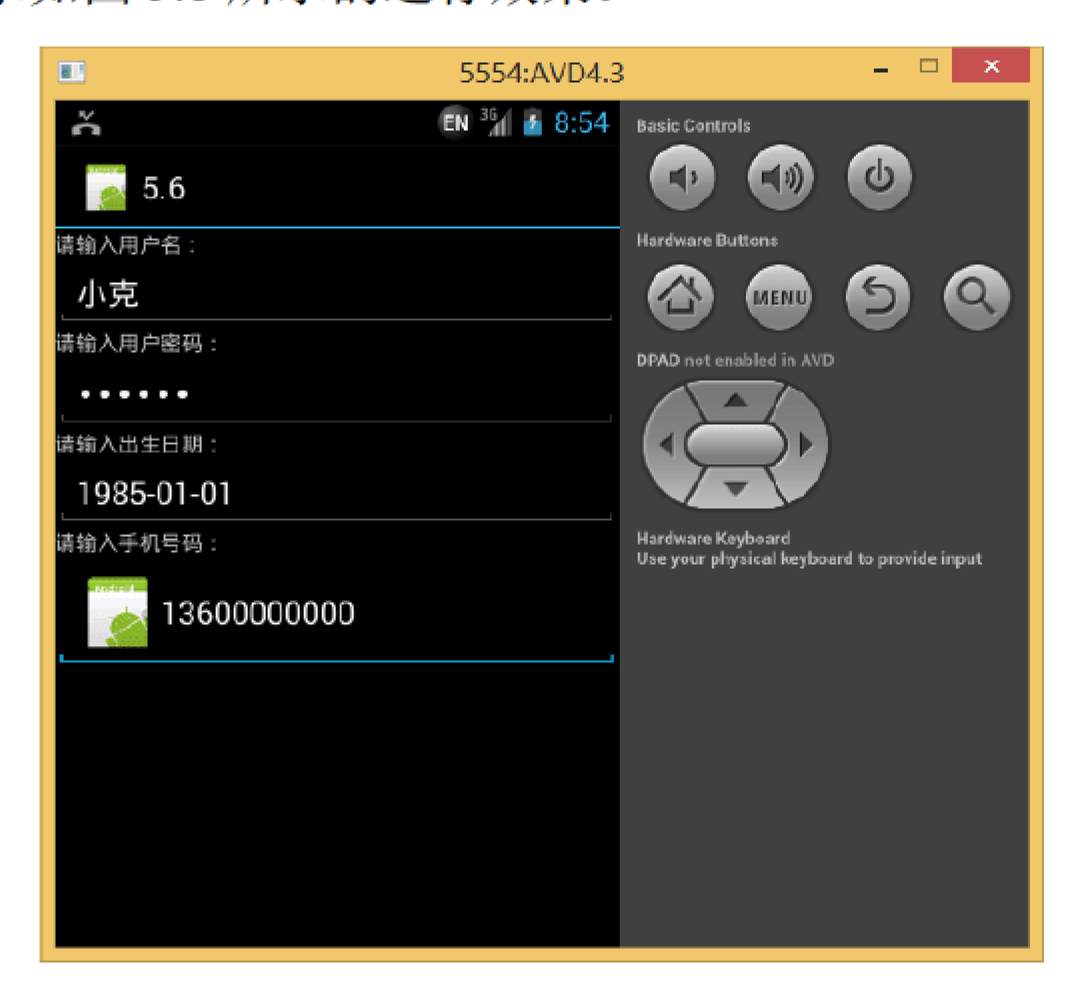
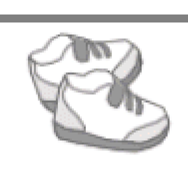
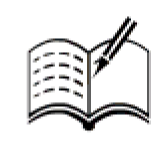


图 5.8 使用 EditText 组件实现用户注册信息的输入

5.2.3 AutoCompleteTextView 组件

AutoCompleteTextView 组件是 Android 中提供的一个自动提示组件，类似于在“百度”中搜





Note

索内容时，当用户在搜索文本框中输入内容时，“百度”会自动提示很多与用户的输入接近的内容供选择。

在 Android 中，可以使用两种方法向屏幕中添加 AutoCompleteTextView 组件：一种是通过在 XML 布局文件中使用<AutoCompleteTextView>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<AutoCompleteTextView>标记在 XML 布局文件中添加。在 XML 布局文件中添加 AutoCompleteTextView 组件的基本语法格式如下：

```
<AutoCompleteTextView
属性列表
/>
```

AutoCompleteTextView 组件继承自 EditText，所以它支持 EditText 组件提供的属性，同时，该组件还支持如表 5.3 所示的 XML 属性。

表 5.3　AutoCompleteTextView 支持的 XML 属性

XML 属性	描　　述
android:completionHint	用于为弹出的下拉菜单指定提示标题
android:completionThreshold	用于指定用户至少输入几个字符才会显示提示
android:dropDownHeight	用于指定下拉菜单的高度
android:dropDownHorizontalOffset	用于指定下拉菜单与文本之间的水平偏移。下拉菜单默认与文本框左对齐
android:dropDownVerticalOffset	用于指定下拉菜单与文本之间的垂直偏移。下拉菜单默认紧跟文本框
android:dropDownWidth	用于指定下拉菜单的宽度
android:popupBackground	用于为下拉菜单设置背景

下面将给出一个关于 AutoCompleteTextView 组件的实例。

【例 5.7】 在 Eclipse 中创建 Android 项目，主要使用 AutoCompleteTextView 组件实现自动提示功能。

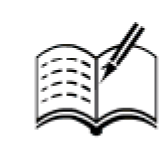
实例位置：光盘\MR\Instance\05\5.7

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的垂直线性布局管理器 LinearLayout 修改为 RelativeLayout，然后在其中添加一个 TextView 组件，用来显示文本信息；添加一个 AutoCompleteTextView 组件，用来输入文本，并显示自动提示列表；添加一个 Button 组件，用来作为“搜索”按钮。其代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    >
     <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
```





```
            android:layout_height="wrap_content"
            android:text="请输入查询关键字"
            android:textSize="24sp"
        />
        <AutoCompleteTextView
            android:id="@+id/actxt"
            android:layout_below="@id/tv"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
        />
        <Button
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_below="@id/actxt"
            android:layout_alignParentRight="true"
            android:text="搜索"
        />
</RelativeLayout>
```



说明：

这里用到了 RelativeLayout 布局方式，RelativeLayout 布局表示相对布局管理器。

（2）打开 MainActivity.java 文件，首先定义一个静态字符串数组，用来存储数据源，然后在 OnCreate()方法中，通过 setAdapter()方法为 AutoCompleteTextView 组件设置 ArrayAdapter 数据源。其代码如下：

```
public class MainActivity extends Activity {
    private static final String[] autoInfo=new String[]
        {"明日科技","C#编程词典","C#从入门到精通","Android 手机","Android 操作系统","Android 实例","Android 项目"};                 //定义字符串数组
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line,autoInfo);                 //创建 ArrayAdapter 对象，并以下拉列表形式进行显示
        //找到布局文件中的 AutoCompleteTextView 组件
        AutoCompleteTextView actxTextView=(AutoCompleteTextView) findViewById(R.id.actxt);
        actxTextView.setAdapter(adapter);   //设置数据源
    }
}
```

运行本实例，当在文本框中输入内容时，程序会自动在数据源中寻找是否有与输入相匹配的内容，如果有，则以列表形式显示出来，如图 5.9 所示。



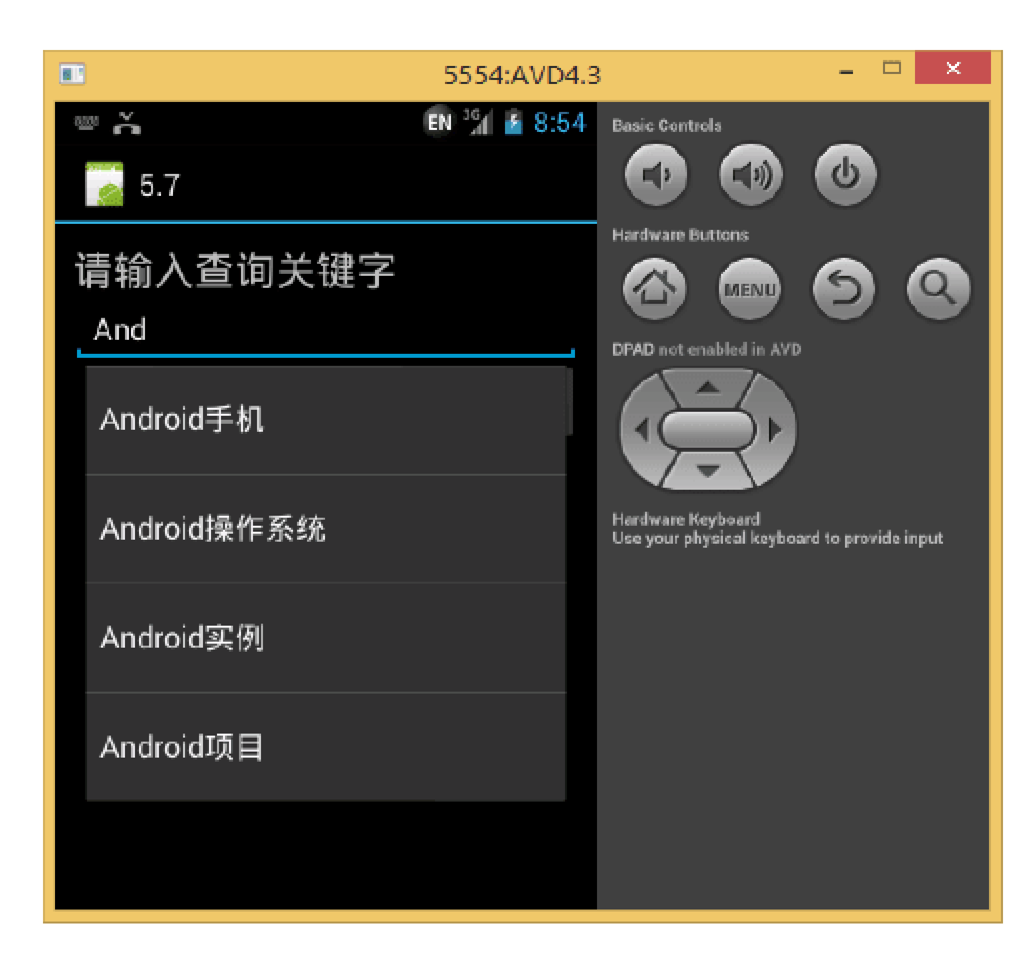



图 5.9　使用 AutoCompleteTextView 组件实现自动提示功能

5.3　按钮类组件

Android 中提供了两种按钮组件：一种是普通按钮，另一种是图片按钮，这两种按钮都是用于在 UI 界面上生成一个可以单击的按钮。当用户单击按钮时，将会触发一个 onClick 事件，可以通过为按钮添加单击事件监听指定所要触的动作。下面将对普通按钮和图片按钮进行详细介绍。

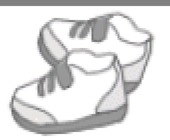
5.3.1　Button 组件

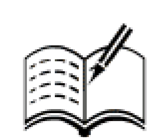
在 Android 中，可以使用两种方法向屏幕中添加按钮：一种是通过在 XML 布局文件中使用<Button>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<Button>标记在 XML 布局文件中添加。在 XML 布局文件中添加普通按钮的基本格式如下：

```
<Button
android:text="显示文本"
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/ >
```

在屏幕上添加按钮后，还需要为按钮添加单击事件监听，才能让按钮发挥其特有的用途。在 Android 中，提供了两种为按钮添加单击事件监听的方法，一种是可以在 Java 代码中完成，例如，在 Activity 的 onCreate()方法中完成，其具体代码如下：

```
import android.view.View.OnClickListener;
import android.widget.Button;
```






```
Button login=(Button)findViewById(R.id.login);          //通过 id 获取布局文件中添加的按钮
login.setOnClickListener(new OnClickListener() {        //为按钮添加单击事件监听
        @Override
        public void onClick(View v) {
                //编写要执行的动作代码
        }
});
```

另一种是在 Activity 中编写一个包含一个 View 类型参数的方法，并且将要触发的动作代码放在该方法中，然后在布局文件中，通过 android:onClick 属性指定对应的方法名实现。例如，在 Activity 中编写了一个 myClick()方法，其关键代码如下：

```
public void myClick(View view){
        //编写要执行的动作代码
}
```

这时就可以在布局文件中通过“android:onClick="myClick"”为按钮添加单击事件监听。

下面将给出一个关于 Button 按钮的实例。

【例 5.8】 在 Eclipse 中创建 Android 项目，主要实现添加按钮和为其设置单击监听事件的功能。

实例位置：光盘\MR\Instance\05\5.8

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加两个 Butotn 组件，即 btn1 和 btn2，并分别设置它们的文本为“按钮一”和“按钮二”。其代码如下：

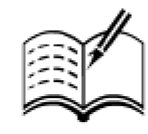
```
<Button
    android:id="@+id/btn1"
    android:text="按钮一"
    android:layout_width="84dp"
    android:layout_height="wrap_content"
    />
    <Button
    android:id="@+id/btn2"
    android:layout_height="wrap_content"
    android:text="按钮二"
    android:layout_width="84dp"
    />
```

（2）打开 MainActivity.java 文件，在 OnCreate()方法中，获取布局文件中的 Button 按钮，并分别为它们设置单击监听事件。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button1=(Button) findViewById(R.id.btn1);      //获取布局文件中的 btn1 按钮
        Button button2=(Button) findViewById(R.id.btn2);      //获取布局文件中的 btn2 按钮
```



```
        button1.setOnClickListener(listener);          //为 btn1 按钮添加监听事件
        button2.setOnClickListener(listener);          //为 btn2 按钮添加监听事件
    }
```



Note

（3）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 OnClick()方法，在该方法中，根据单击的 Button 组件 id，弹出相应的信息提示框。其代码如下：

```
private OnClickListener   listener=new OnClickListener() {         //创建监听事件
    public void onClick(View v)
        Button btnButton=(Button) v;                               //将 View 对象强制转换为 Button 对象
        switch (btnButton.getId()) {                               //获取 Button 对象的 id
        case R.id.btn1:                                            //如果是 btn1 按钮
        //显示相应的对话框
        Toast.makeText(MainActivity.this, "按钮一的单击事件", Toast.LENGTH_LONG).show();
        break;
        case R.id.btn2:                                            //如果是 btn2 按钮
        Toast.makeText(MainActivity.this, "按钮二的单击事件", Toast.LENGTH_SHORT).show();
        break;
        }
    }
};
```

运行本实例，将显示如图 5.10 所示的运行效果，单击“按钮一”按钮，将显示“按钮一的单击事件”提示信息，单击“按钮二”按钮，将显示“按钮二的单击事件”提示信息。

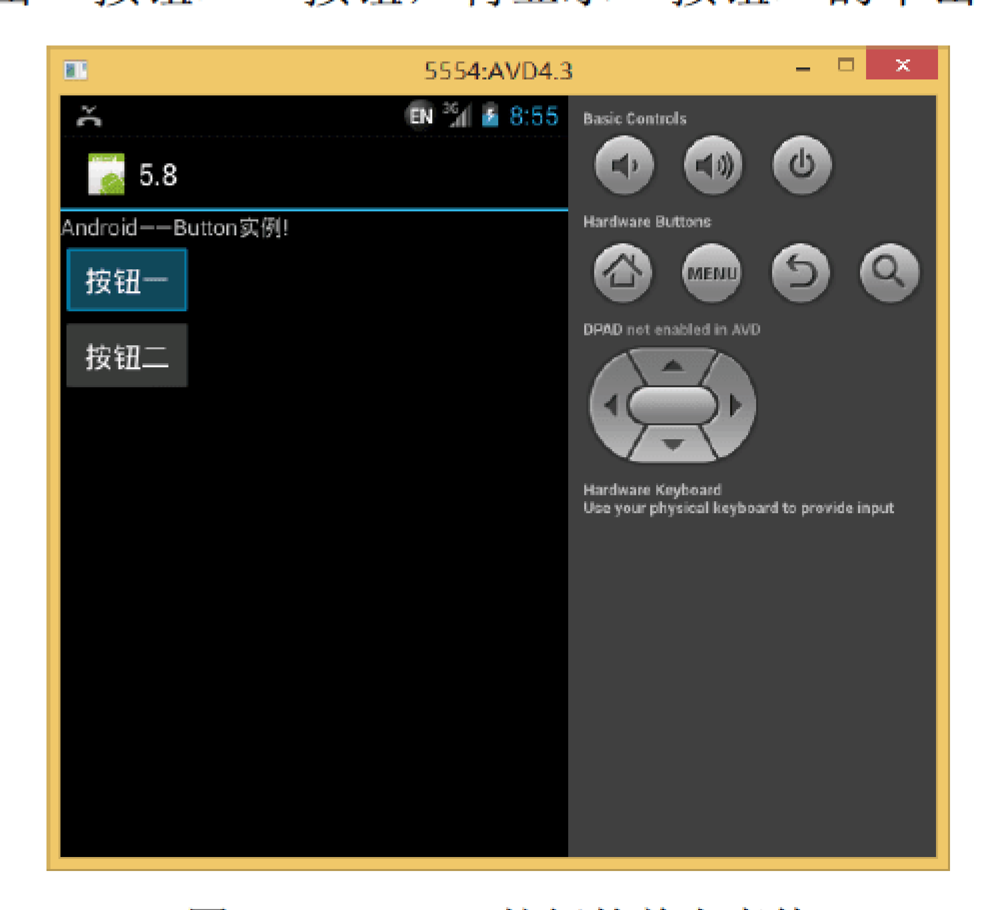


图 5.10　Button 按钮的单击事件

5.3.2　ImageButton 组件

图片按钮与普通按钮的使用方法基本相同，只不过图片按钮使用 ImageButton 关键字定义，并且还可以为其指定 android:src 属性，该属性用于设置要显示的图片。在布局文件中，添加图片按钮的基本格式如下：



```
<ImageButton
      android:id="@+id/imageButton1"
      android:src="@drawable/图片文件名"
      android:background="#000"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
</ImageButton>
```



Note

同普通按钮一样，图片按钮也需要为其添加单击事件监听，具体方法同普通按钮是相同的，这里将不再赘述。下面将给出一个关于图片按钮的实例。

【例 5.9】 在 Eclipse 中创建 Android 项目，主要实现添加图片按钮的功能。

实例位置：光盘\MR\Instance\05\5.9

修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 ImageButton 组件，并为其设置背景图片。其代码如下：

```
<ImageButton
      android:id="@+id/login"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/login"
      android:background="#000"
/>
```

说明：

ImageButton 按钮的监听事件的设置与 Button 按钮类似，详请参见 5.3.1 节。

运行本实例，将显示如图 5.11 所示的运行效果。

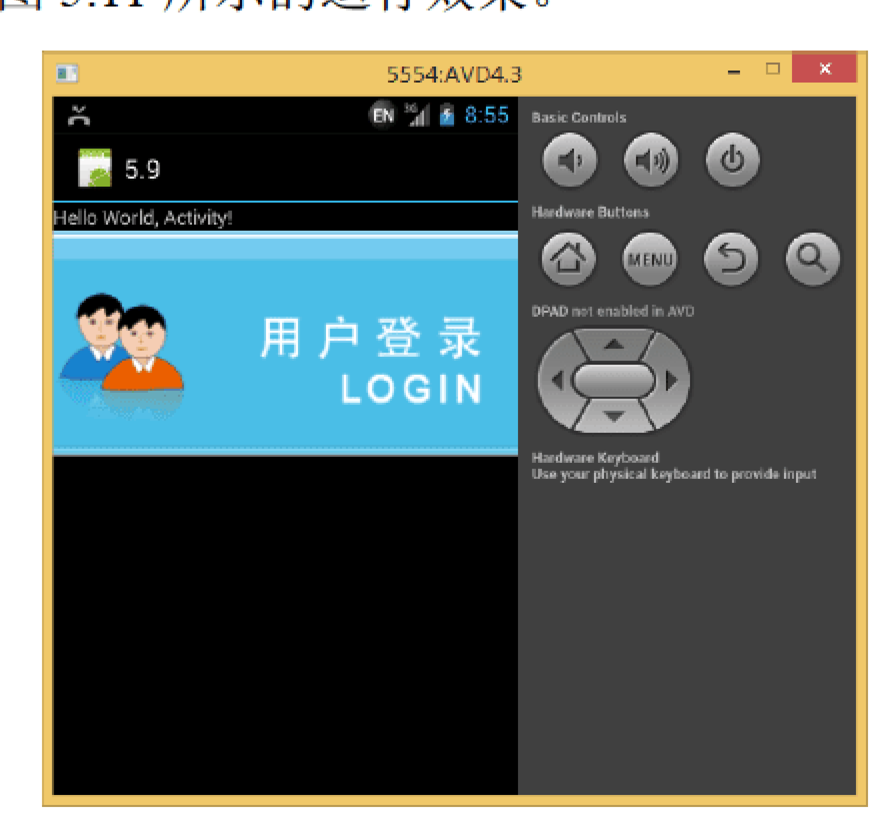


图 5.11　图片按钮的使用

5.3.3　ToggleButton 组件

ToggleButton 组件是 Android 中提供的一种特殊的按钮控件，在 Android 中，可以使用两种方



法向屏幕中添加 ToggleButton 组件：一种是通过在 XML 布局文件中使用<ToggleButton>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<ToggleButton>标记在 XML 布局文件中添加。在 XML 布局文件中添加编辑框的基本语法格式如下：



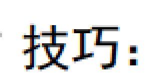
Note

```
< ToggleButton
属性列表
/>
```

ToggleButton 组件继承自 Button，所以它支持 Button 组件提供的属性，同时，该组件还支持如表 5.4 所示的 XML 属性。

表 5.4　ToggleButton 支持的 XML 属性

XML 属性	描　　述
android:textOn	设置控件在选中时显示的文本
android:textOff	设置控件在未选中时显示的文本



技巧：

在 Java 源文件中，开发人员可以使用 ToggleButton 对象的 getTextOn()方法获取 ToggleButton 组件选中时显示的文本，使用 ToggleButton 对象的 getTextOff()方法获取 ToggleButton 组件未选中时显示的文本；同理，可以使用 ToggleButton 对象的 getText()方法获取 ToggleButton 组件当前显示的文本。

下面将给出一个关于 ToggleButton 组件的实例。

【例 5.10】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中添加 ToggleButton 组件并获取其当前文本的功能。

实例位置：光盘\MR\Instance\05\5.10

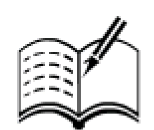
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 ToggleButton 组件，并分别设置其 textOn 和 textOff 属性为“开始”和“停止”。其代码如下：

```
<ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="178dp"
    android:layout_height="wrap_content"
    android:textOn="开始"
    android:textOff="停止"
/>
```

（2）打开 MainActivity.java 文件，首先根据 id 获取布局文件中的 ToggleButton 组件，然后为该组件设置单击监听事件，在监听事件中，实现在对话框中显示当前 ToggleButton 组件中文本的功能。其代码如下：





Note

```
//获取布局文件中的 ToggleButton 组件
ToggleButton tButton=(ToggleButton) findViewById(R.id.toggleButton1);
tButton.setOnClickListener(new OnClickListener() {          //设置监听事件
        @Override
        public void onClick(View v) {
              //TODO Auto-generated method stub
              ToggleButton tButton1=(ToggleButton) v;          //创建 ToggleButton 组件
              Toast.makeText(MainActivity.this, tButton1.getText(), Toast.LENGTH_SHORT).show();
                                                               //弹出当前 ToggleButton 组件的文本
        }
});
```

运行本实例，当用户单击 ToggleButton 按钮时，在弹出的对话框中显示当前 ToggleButton 按钮上的文本，如图 5.12 所示。

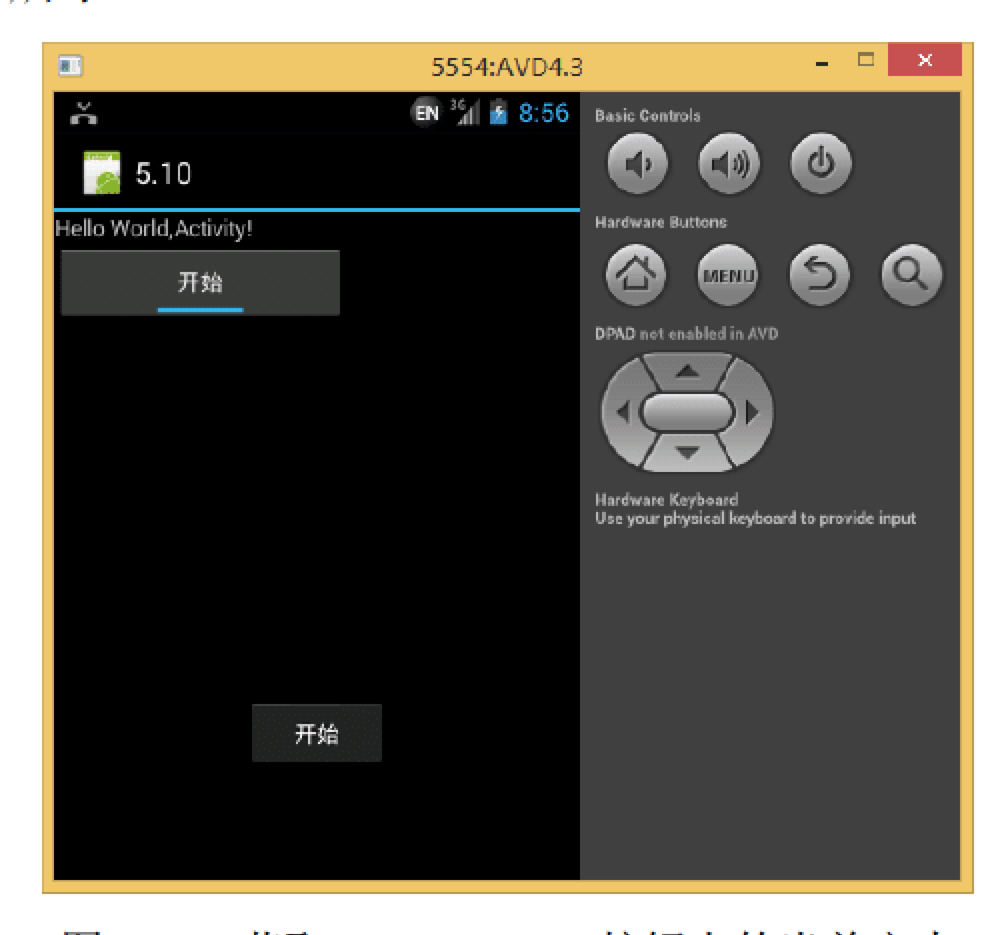


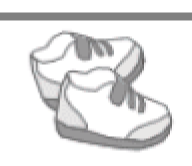
图 5.12　获取 ToggleButton 按钮上的当前文本

5.4　选择类组件

在 Android 中，单选按钮和复选按钮都继承了普通按钮，因此，它们都可以直接使用普通按钮支持的各种属性和方法。与普通按钮不同的是，它们提供了可选中的功能。下面将对单选按钮和复选按钮进行详细介绍。

5.4.1　RadioButton 组件

在默认的情况下，单选按钮显示一个圆形图标，并且在该图标旁边放置一些说明性文字，而在程序中，一般将多个单选按钮放置在按钮组中，使这些单选按钮表现出某种功能，当用户选中某个单选按钮后，按钮组中的其他单选按钮将被自动取消选取状态。在 Android 中，单选按钮使用 RadioButton 表示，而 RadioButton 类又是 Button 的子类，所以单选按钮可以直接使用 Button



支持的各种属性。

在 Android 中，可以使用两种方法向屏幕中添加单选按钮：一种是通过在 XML 布局文件中使用<RadioButton>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<RadioButton>标记在 XML 布局文件中添加。在 XML 布局文件中添加单选按钮的基本格式如下：

```
<RadioButton
    android:text="显示文本"
    android:id="@+id/ID 号"
    android:checked="true|false"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

RadioButton 组件的 android:checked 属性用于指定选中状态，属性值为 true 时，表示选中，属性值为 false 时，表示不选中，默认为 false。

通常情况下，RadioButton 组件需要与 RadioGroup 组件一起使用，组成一个单选按钮组。在 XML 布局文件中，添加 RadioGroup 组件的基本格式如下：

```
<RadioGroup
    android:id="@+id/radioGroup1"
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <!-- 添加多个 RadioButton 组件 -->
</RadioGroup>
```

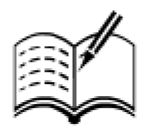
【例 5.11】 在 Eclipse 中创建 Android 项目，实现在屏幕上添加选择性别的单选按钮组。



修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的垂直线性布局管理器设置为水平布局管理器，在该布局管理器中添加一个 TextView、一个包含两个单选按钮的单选按钮组和一个“提交”按钮。其具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="性别："
    android:height="50px" />
<RadioGroup
    android:id="@+id/radioGroup1"
```





```
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton
            android:layout_height="wrap_content"
            android:id="@+id/radio0"
            android:text="男"
            android:layout_width="wrap_content"
            android:checked="true"/>
        <RadioButton
            android:layout_height="wrap_content"
            android:id="@+id/radio1"
            android:text="女"
            android:layout_width="wrap_content"/>
    </RadioGroup>
        <Button
            android:text="提交"
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
        />
</LinearLayout>
```

运行本实例，将显示如图 5.13 所示的运行效果。

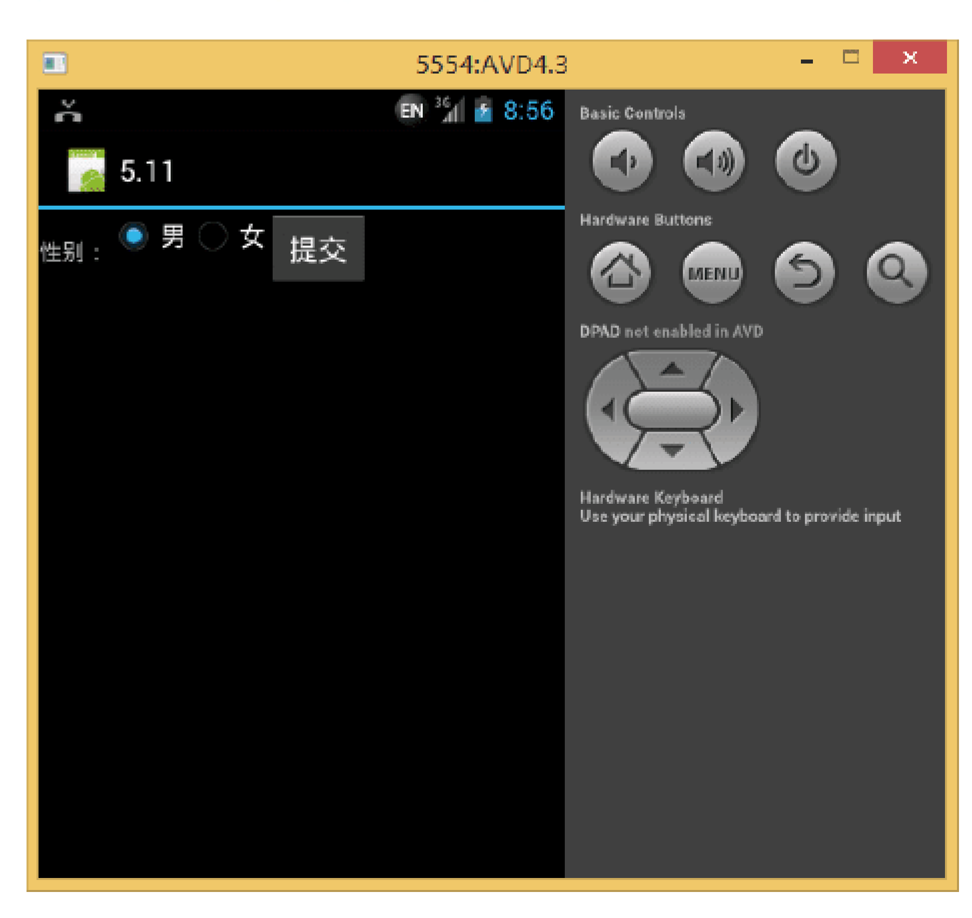
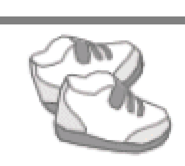


图 5.13　添加选择性别的单选按钮

在屏幕中添加单选按钮组后，还需要获取单选按钮组中选中项的值。在获取单选按钮组中选中项的值时，通常存在以下两种情况：一种是在改变单选按钮组的值时获取，另一种是在单击其他按钮时获取。下面分别介绍这两种情况所对应的实现方法。

☑　在改变单选按钮组的值时获取

在改变单选按钮组的值而获取选中项的值时，首先需要获取单选按钮组，然后为其添加 OnCheckedChangeListener 监听事件，并在其 onCheckedChanged()方法中根据参数 checkedId 获取被选中的单选按钮，然后通过其 getText()方法获取该单选按钮对应的值。例如，要获取 id 属性



为 radioGroup1 的单选按钮组的值，可以通过下面的代码实现。

```
RadioGroup sex=(RadioGroup)findViewById(R.id.radioGroup1);
sex.setOnCheckedChangeListener(new OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(RadioGroup group, int checkedId) {
            RadioButton r=(RadioButton)findViewById(checkedId);
            r.getText();                                         //获取被选中的单选按钮的值
      }
});
```

☑ 单击其他按钮时获取

单击其他按钮而获取选中项的值时，首先需要在该按钮的单击事件监听的 onClick()方法中，通过 for 循环语句遍历当前单选按钮组，并根据被遍历到的单选按钮的 isChecked()方法判断该按钮是否被选中，如果被选中，通过单选按钮的 getText()方法获取对应的值。例如，要在单击“提交”按钮时，获取 id 属性为 radioGroup1 的单选按钮组的值，可以通过下面的代码实现。

```
final RadioGroup sex=(RadioGroup)findViewById(R.id.radioGroup1);
Button button=(Button)findViewById(R.id.button1);          //获取一个提交按钮
button.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
      for(int i=0;i<sex.getChildCount();i++){
            RadioButton r=(RadioButton)sex.getChildAt(i);      //根据索引值获取单选按钮
            if(r.isChecked()){                                 //判断单选按钮是否被选中
                  r.getText();                                 //获取被选中的单选按钮的值
                  break;                                       //跳出 for 循环
            }
      }
      }
});
```

5.4.2 CheckBox 组件

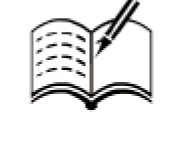
在默认情况下，复选按钮显示一个方块图标，并且在该图标旁边放置一些说明性文字。与单选按钮唯一不同的是，复选按钮可以进行多选设置，每一个复选按钮都提供“选中”和“不选中”两种状态。在 Android 中，复选按钮使用 CheckBox 表示，而 CheckBox 类又是 Button 的子类，所以复选按钮可以直接使用 Button 支持的各种属性。

在 Android 中，可以使用两种方法向屏幕中添加复选按钮：一种是通过在 XML 布局文件中使用<CheckBox>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<CheckBox>标记在 XML 布局文件中添加。在 XML 布局文件中添加复选按钮的基本格式如下：

```
<CheckBox android:text="显示文本"
      android:id="@+id/ID 号"
```



```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
/>
```



由于复选框可以选中多项，所以为了确定用户是否选择了某一项，还需要为每一个选项添加 setOnCheckedChangeListener 事件监听。例如，要为 id 为 like1 的复选按钮添加状态改变事件监听，可以使用下面的代码。

```
final CheckBox like1=(CheckBox)findViewById(R.id.like1);            //根据 id 属性获取复选按钮
like1.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(like1.isChecked()){                                       //判断该复选按钮是否被选中
            like1.getText();                                         //获取选中项的值
        }
    }
});
```

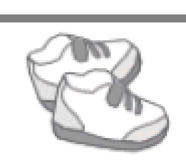
【例 5.12】 在 Eclipse 中创建 Android 项目，实现在屏幕上添加选择爱好的复选按钮，并获取选中的值。

实例位置：光盘\MR\Instance\05\5.12

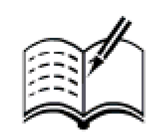
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认的线性布局管理器中添加 1 个 TextView、3 个复选按钮和 1 个“提交”按钮。关键代码如下：

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="爱好："
    android:width="50px"
    android:gravity="right"
    android:height="30px" />
<CheckBox android:text="体育"
    android:id="@+id/like1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<CheckBox android:text="音乐"
    android:id="@+id/like2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<CheckBox android:text="美术"
    android:id="@+id/like3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<Button
    android:text="提交"
    android:id="@+id/button1"
```



```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```



Note

（2）在主活动中创建一个 OnCheckedChangeListener 对象，在创建该对象时，重写 onCheckedChanged()方法，当复选按钮被选中时，输出一条日志信息，显示被选中的复选按钮。其具体代码如下：

```
//创建一个状态改变监听对象
private OnCheckedChangeListener checkBox_listener=new OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(isChecked){          //判断复选按钮是否被选中
            Log.i("复选按钮","选中了["+buttonView.getText().toString()+"]");
        }
    }
};
```

（3）在主活动的 onCreate()方法中获取添加的 3 个复选按钮，并为每个复选按钮添加状态改变事件监听。其关键代码如下：

```
final CheckBox like1=(CheckBox)findViewById(R.id.like1);          //获取第 1 个复选按钮
final CheckBox like2=(CheckBox)findViewById(R.id.like2);          //获取第 2 个复选按钮
final CheckBox like3=(CheckBox)findViewById(R.id.like3);          //获取第 3 个复选按钮
like1.setOnCheckedChangeListener(checkBox_listener);              //为 like1 添加状态改变监听
like2.setOnCheckedChangeListener(checkBox_listener);              //为 like2 添加状态改变监听
like3.setOnCheckedChangeListener(checkBox_listener);              //为 like3 添加状态改变监听
```

（4）获取“提交”按钮，并为其添加单击事件监听，在该事件监听的 onClick()方法中通过 if 语句获取被选中的复选按钮的值，并通过一个提示信息框显示。其具体代码如下：

```
Button button=(Button) findViewById(R.id.button1);               //获取“提交”按钮
//为“提交”按钮添加单击事件监听
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        String like="";                                           //保存选中的值
        if(like1.isChecked())                                     //当第 1 个复选按钮被选中
        like+=like1.getText().toString()+" ";
        if(like2.isChecked())                                     //当第 2 个复选按钮被选中
            like+=like2.getText().toString()+" ";
        if(like3.isChecked())                                     //当第 3 个复选按钮被选中
            like+=like3.getText().toString()+" ";
        //显示被选中的复选按钮
        Toast.makeText(MainActivity.this, like, Toast.LENGTH_SHORT).show();
    }
});
```

运行本实例，将显示 3 个用于选取爱好的复选按钮，当用户将其中的某个复选框选中时，单击“提交”按钮，将在弹出的提示框中显示用户选中的信息，如图 5.14 所示。



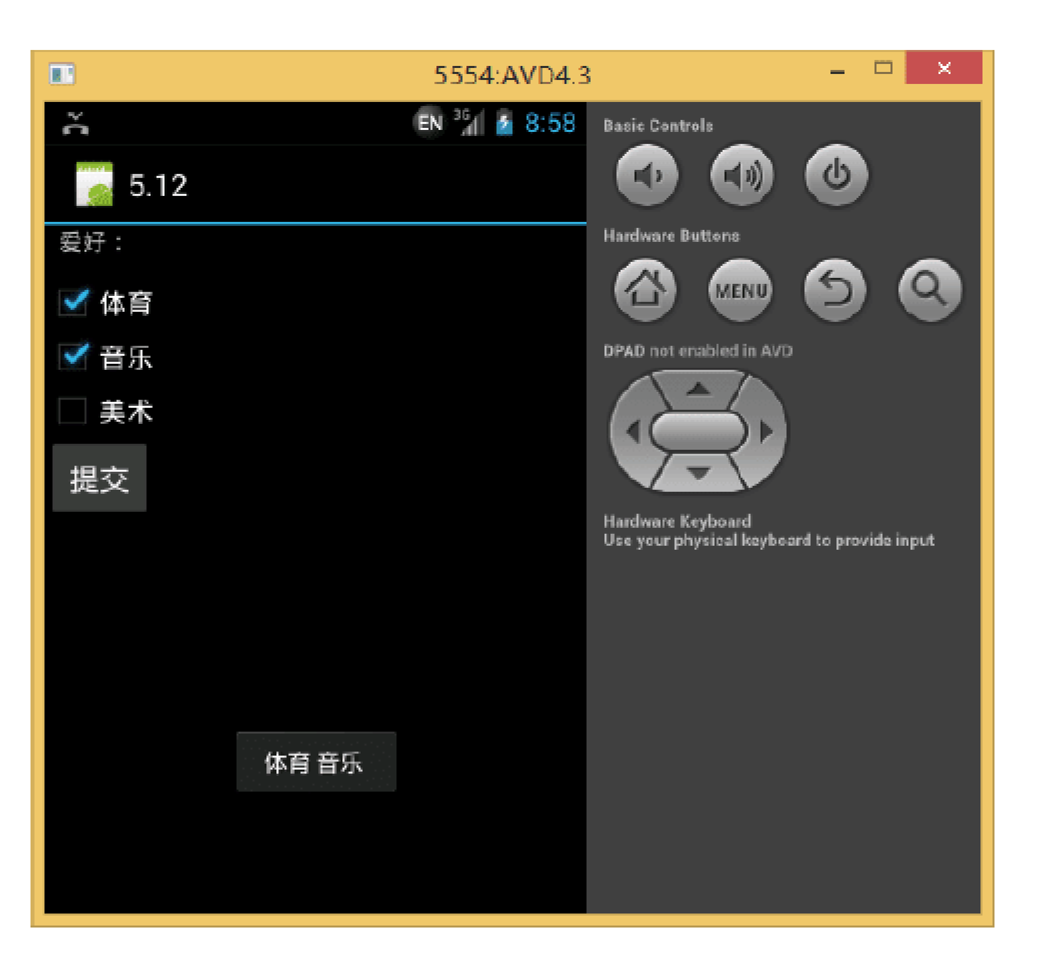


图 5.14　使用复选按钮实现用户爱好的选择

Note

5.5　列表类组件

Android 中提供了两种比较常用的列表类组件，即 ListView 和 Spinner 组件，其中，ListView 组件表示列表选择组件，而 Spinner 组件表示下拉列表选择组件，下面分别对这两种列表组件进行详细介绍。

5.5.1　ListView 组件

列表视图是 Android 中最常用的一种视图组件，它以垂直列表的形式列出需要显示的列表项。例如，显示系统设置项或功能内容列表等。在 Android 中，可以使用两种方法向屏幕中添加列表视图：一种是直接使用 ListView 组件创建；另一种是让 Activity 继承 ListActivity 实现。下面分别进行介绍。

1．直接使用 ListView 组件创建

直接使用 ListView 组件创建列表视图，也可以有两种方式：一种是通过在 XML 布局文件中使用<ListView>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<ListView>标记在 XML 布局文件中添加。在 XML 布局文件中添加 ListView 的基本格式如下：

```
<ListView
属性列表
/>
```

ListView 支持的常用 XML 属性如表 5.5 所示。



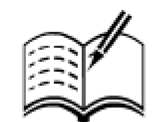


Note

表 5.5　ListView 支持的 XML 属性

XML 属性	描　　述
android:divider	用于为列表视图设置分隔条，既可以用颜色分隔，也可以用 Drawable 资源分隔
android:dividerHeight	用于设置分隔条的高度
android:entries	用于通过数组资源为 ListView 指定列表项
android:footerDividersEnabled	用于设置是否在 footer View 之前绘制分隔条，默认值为 true，设置为 false 时，表示不绘制。使用该属性时，需要通过 ListView 组件提供的 addFooterView()方法为 ListView 设置 footer View
android:headerDividersEnabled	用于设置是否在 header View 之后绘制分隔条，默认值为 true，设置为 false 时，表示不绘制。使用该属性时，需要通过 ListView 组件提供的 addHeaderView()方法为 ListView 设置 header View

【例 5.13】 在 Eclipse 中创建 Android 项目，实现通过数组资源为 ListView 设置列表项的功能。

实例位置：光盘\MR\Instance\05\5.13

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并添加一个 ListView 组件，通过数组资源为其设置列表项。其具体代码如下：

```
<ListView android:id="@+id/listView1"
    android:entries="@array/ctype"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"/>
```

（2）在上面的代码中，使用了名称为 ctype 的数组资源，因此，需要在 res\value 目录的 strings.xml 资源文件中添加名称为 ctype 的字符串数组。其关键代码如下：

```
<resources>
    <string-array name="ctype">
     <item>C#编程词典</item>
     <item>JAVA 编程词典</item>
     <item>VB 编程词典</item>
     <item>VC 编程词典</item>
     <item>ASP 编程词典</item>
     <item>Delphi 编程词典</item>
     <item>ASP.NET 编程词典</item>
    </string-array>
</resources>
```

运行本实例，将显示如图 5.15 所示的列表视图。

在使用列表视图时，重要的是如何设置选项内容。ListView 如果在布局文件中没有为其指定要显示的列表项，也可以通过为其设置 Adapter 来指定需要显示的列表项。通过 Adapter 来为 ListView 指定要显示的列表项，可以分为以下两个步骤。



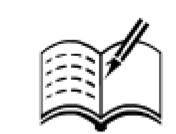




图 5.15　在布局文件中添加的列表视图

（1）创建 Adapter 对象。对于纯文字的列表项，通常使用 ArrayAdapter 对象。创建 ArrayAdapter 对象通常可以有两种情况，一种是通过数组资源文件创建，另一种是通过在 Java 文件中使用字符串数组创建，它们的创建方式分别如下。

☑　通过数组资源文件创建

通过数组资源文件创建适配器，需要使用 ArrayAdapter 类的 createFromResource()方法。其具体代码如下：

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
          this, R.array.ctype,android.R.layout. simple_list_item_checked);          //创建一个适配器
```

☑　通过在 Java 文件中使用字符串数组创建

通过在 Java 文件中使用字符串数组创建适配器，首先需要创建一个一维的字符串数组，用于保存要显示的列表项，然后使用 ArrayAdapter 类的构造方法 ArrayAdapter(Context context, int textViewResourceId, T[] objects)创建一个 ArrayAdapter 类的对象。其具体代码如下：

```
String[] ctype=new String[]{"身份证","学生证","军人证"};
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,android.R.layout. simple_list_item_checked, ctype);
```

这里需要注意的是，在创建 ArrayAdapter 对象时，需要指定列表项的外观形式。为 ListView 指定的外观形式通常有以下 5 种。

☑　simple_list_item_1：每个列表项都是一个普通的文本。
☑　simple_list_item_2：每个列表项都是一个普通的文本（字体略大）。
☑　simple_list_item_checked：每个列表项都有一个已勾选的列表项。
☑　simple_list_item_multiple_choice：每个列表项都是带多选框的文本。
☑　simple_list_item_single_choice：每个列表项都是带单选按钮的文本。

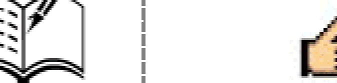
（2）将创建的适配器对象与 ListView 相关联，可以通过 ListView 对象的 setAdapter()方法实现。其具体代码如下：

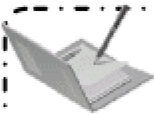
```
listView.setAdapter(adapter);                              //将适配器与 ListView 关联
```



下面通过一个具体的实例来演示如何使用适配器指定列表项的方式创建 ListView。

【例 5.14】 在 Eclipse 中创建 Android 项目，实现在屏幕中添加列表视图，并为其设置 footer view 和 header view 的功能。





实例位置：光盘\MR\Instance\05\5.14

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并添加一个 ListView 组件。添加 ListView 组件的布局代码如下：

```
<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:divider="@drawable/icon"
    android:dividerHeight="3px"/>
```

说明：

在上面的代码中，为 ListView 组件设置了作为分隔图的图像，以及分隔符的高度。

（2）在主活动的 onCreate()方法中为 ListView 组件创建并关联适配器。首先获取布局文件中添加的 ListView，然后创建适配器，并将其与 ListView 相关联。其关键代码如下：

```
ListView listView=(ListView) findViewById(R.id.listView1);  //获取 listView1 组件
/****************创建用于为 ListView 指定列表项的适配器*******************/
String[] ctype=new String[]{"C#编程词典","JAVA 编程词典","VB 编程词典","VC 编程词典","ASP.NET 编程词典"};
ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,android.R.layout. simple_list_item_ checked,
ctype);
/*************************************************************************/
listView.setAdapter(adapter);                                    //将适配器与 ListView 关联
```

（3）为了在单击 ListView 的各列表项时获取选择项的值，需要为 ListView 添加 OnItemClick Listener 事件监听。其具体代码如下：

```
listView.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View arg1, int pos,  long id) {
    String result = parent.getItemAtPosition(pos).toString();    //获取选择项的值
    Toast.makeText(MainActivity.this, result, Toast.LENGTH_SHORT).show();
    }
});
```

运行本实例，将显示如图 5.16 所示的运行效果。

2. 让 Activity 继承 ListActivity 实现

如果程序的窗口仅仅需要显示一个列表，则可以直接让 Activity 继承 ListActivity 来实现。继承了 ListActivity 的类中无须调用 setContentView()方法来显示页面，而是可以直接为其设置适



Note

配器，从而显示一个列表。下面通过一个实例来说明如何通过继承 ListActivity 实现列表。

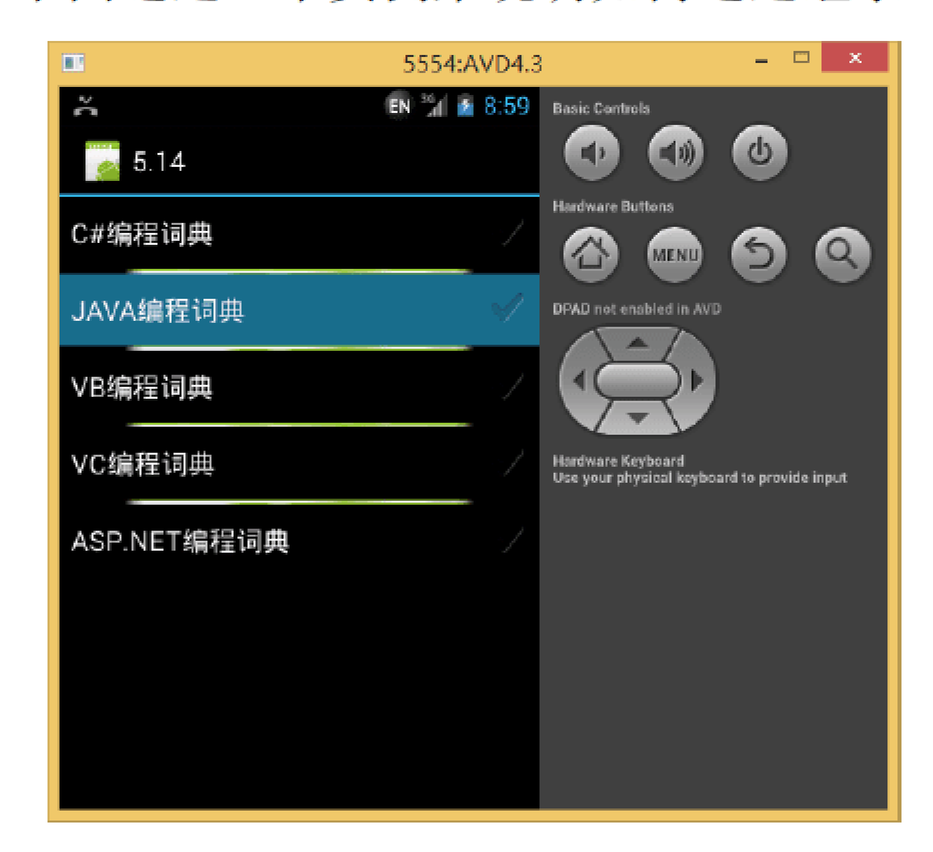




图 5.16　使用适配器为 ListView 设置列表项

【例 5.15】 在 Eclipse 中创建 Android 项目，通过在 Activity 中继承 ListActivity 实现列表。

实例位置：光盘\MR\Instance\05\5.15

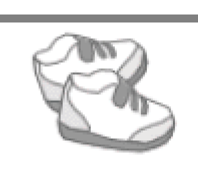
程序的开发步骤如下：

（1）将新建项目中的主活动 MainActivity 修改为继承 ListActivity 的类，并将默认的设置用户布局的代码删除，然后在 onCreate()方法中创建作为列表项的 Adapter，并且使用 setListAdapter()方法将其添加到列表中。其关键代码如下：

```
public class MainActivity extends ListActivity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		/****************创建用于为 ListView 指定列表项的适配器*******************/
		String[] ctype=new String[]{"C#编程词典","JAVA 编程词典","VB 编程词典","VC 编程词典",
"ASP.NET 编程词典"};
		ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
		android.R.layout.simple_list_item_single_choice,ctype);
		/*****************************************************************************/
		setListAdapter(adapter);				//设置该窗口中显示的列表
	}
}
```

（2）为了在单击 ListView 的各列表项时获取选择项的值，需要重写父类中的 onListItemClick()方法。其具体代码如下：

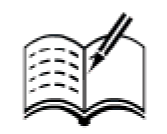
```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
	super.onListItemClick(l, v, position, id);
		String result = l.getItemAtPosition(position).toString();	//获取选择项的值
			Toast.makeText(MainActivity.this, result, Toast.LENGTH_SHORT).show();
		}
```





说明：

本实例的运行效果与例 5.14 的效果类似，详请参见图 5.16。



Note

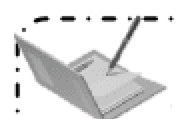
5.5.2 Spinner 组件

Android 中提供的 Spinner 列表选择框相当于在网页中常见的下拉列表框，通常用于提供一系列可选择的列表项，供用户进行选择，从而方便用户。

在 Android 中，可以使用两种方法向屏幕中添加列表选择框：一种是通过在 XML 布局文件中使用<Spinner>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<Spinner>标记在 XML 布局文件中添加。在 XML 布局文件中添加列表选择框的基本格式如下：

```
<Spinner
    android:prompt="@string/info"
    android:entries="@array/数组名称"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:id="@+id/ID 号"
/>
```

其中，android:entries 为可选属性，用于指定列表项，如果在布局文件中不指定该属性，可以在 Java 代码中通过为其指定适配器的方式指定；android:prompt 属性也是可选属性，用于指定列表选择框的标题。

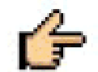


说明：

在 Android 4.3 中，采用默认的主题（Theme.Holo）时，android:prompt 属性看不到具体的效果，但是如果采用 Theme.Black 时，就可以看到在弹出的下拉框上将显示该标题。

通常情况下，如果列表选择框中要显示的列表项是可知的，那么会将其保存在数组资源文件中，然后通过数组资源来为列表选择框指定列表项。这样，就可以在不编写 Java 代码的情况下实现一个列表选择框。下面将通过一个具体的实例来说明如何在不编写 Java 代码的情况下，在屏幕中添加列表选择框。

【例 5.16】 在 Eclipse 中创建 Android 项目，实现在屏幕中添加列表选择框，并获取列表选择框的选择项值的功能。

 **实例位置：光盘\MR\Instance\05\5.16**

程序的开发步骤如下：

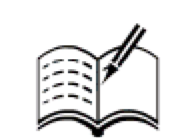
（1）在布局文件中添加一个 Spinner 组件，并为其指定 android:entries。其具体代码如下：

```
<Spinner
    android:entries="@array/ctype"
    android:layout_height="wrap_content"
```



```
        android:layout_width="wrap_content"
        android:id="@+id/spinner1"/>
```

（2）在上面的代码中，使用了名称为 ctype 的数组资源，因此，需要在 res\value 目录的 strings.xml 资源文件中添加名称为 ctype 的字符串数组。其关键代码如下：



Note

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="ctype">
     <item>身份证</item>
     <item>学生证</item>
     <item>军人证</item>
     <item>工作证</item>
     <item>其他</item>
    </string-array>
</resources>
```

（3）这样，就可以在屏幕中添加一个列表选择框。在屏幕上添加列表选择框后，可以使用列表选择框的 getSelectedItem()方法获取列表选择框的选中值。例如，要获取列表选择框的选中项的值，可以使用下面的代码。

```
Spinner spinner = (Spinner) findViewById(R.id.spinner1);                //获取 spinner1 组件
spinner.getSelectedItem();
```

（4）添加列表选择框后，如果需要在用户选择不同的列表项后执行相应的处理，则可以为该列表选择框添加 OnItemSelectedListener 事件监听。例如，为 Spinner 添加选择列表项事件监听，并在 onItemSelected()方法中获取选择项的值输出到日志中，可以使用下面的代码。

```
//为选择列表框添加 OnItemSelectedListener 事件监听
spinner.setOnItemSelectedListener(new OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View arg1,
        int pos, long id) {
        String result = parent.getItemAtPosition(pos).toString();     //获取选择项的值
        Toast.makeText(MainActivity.this, result, Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
    }
});
```

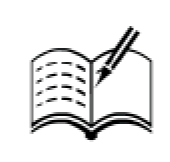
运行本实例，将显示如图 5.17 所示的运行效果。

在使用列表选择框时，如果不在布局文件中直接为其指定要显示的列表框，也可以通过为其指定适配器的方式指定。下面还以例 5.16 为例介绍通过指定适配器的方式指定列表项的方法。

为列表选择框指定适配器，通常分为以下 3 个步骤实现。

（1）创建一个适配器对象，通常使用 ArrayAdapter 类。在 Android 中，创建适配器，通常可以有以下两种情况，一种是通过数组资源文件创建，另一种是通过在 Java 文件中使用字符串





Note

数组创建，这与 5.5.1 节 ListView 组件中介绍的创建 ArrayAdapter 对象基本相同。

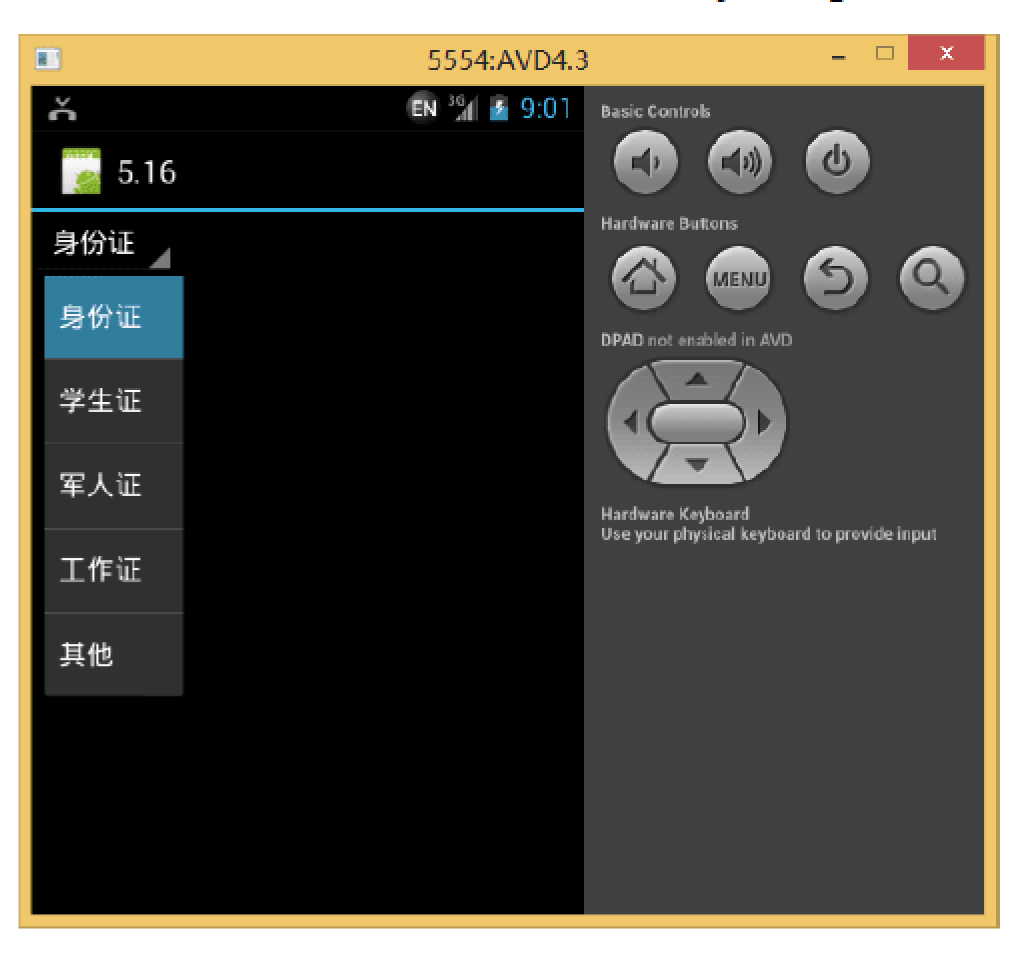


图 5.17　显示列表选择框并获取其选择项

（2）为适配器设置列表框下拉时的选项样式，其具体代码如下：

```
//为适配器设置列表框下拉时的选项样式
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

（3）将适配器与选择列表框关联，其具体代码如下：

```
spinner.setAdapter(adapter);                                    //将适配器与选择列表框关联
```

5.6　图像类组件

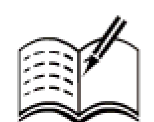
Android 中提供了 3 种常用的图像类组件，分别是 ImageView、Gallery 和 ImageSwitcher 组件，其中，ImageView 组件用来显示图像视图，Gallery 组件用来显示画廊视图，而 ImageSwitcher 组件表示一个图像切换器，下面分别对这 3 种图像类组件进行详细介绍。

5.6.1　ImageView 组件

ImageView 也就是图像视图，用于在屏幕中显示任何的 Drawable 对象，通常用来显示图片。在 Android 中，可以使用两种方法向屏幕中添加图像视图：一种是通过在 XML 布局文件中使用 <ImageView>标记添加；另一种是在 Java 文件中，通过 new 关键字创建出来。推荐采用第一种方法。

在使用 ImageView 组件显示图像时，通常可以将要显示的图片放置在 res\drawable 目录中，然后应用下面的代码将其显示在布局管理器中。





```
<ImageView
    属性列表
    />
```

ImageView 支持的常用 XML 属性如表 5.6 所示。

表 5.6　ImageView 支持的 XML 属性

XML 属性	描　述
android:adjustViewBounds	用于设置 ImageView 是否调整自己的边界来保持所显示图片的长宽比
android:maxHeight	设置 ImageView 的最大高度，需要设置 android:adjustViewBounds 属性值为 true，否则不起作用
android:maxWidth	设置 ImageView 的最大宽度，需要设置 android:adjustViewBounds 属性值为 true，否则不起作用
android:scaleType	用于设置所显示的图片如何缩放或移动以适应 ImageView 的大小，其属性值可以是 matrix（使用 matrix 方式进行缩放）、fitXY（对图片横向、纵向独立缩放，使得该图片完全适应于该 ImageView，图片的纵横比可能会改变）、fitStart（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的左上角）、fitCenter（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的中央）、fitEnd（保持纵横比缩放图片，直到该图片能完全显示在 ImageView 中，缩放完成后该图片放在 ImageView 的右下角）、center（把图像放在 ImageView 的中间，但不进行任何缩放）、centerCrop（保持纵横比缩放图片，以使得图片能完全覆盖 ImageView）或 centerInside（保持纵横比缩放图片，以使得 ImageView 能完全显示该图片）
android:src	用于设置 ImageView 所显示的 Drawable 对象的 ID，例如，设置显示保存在 res\drawable 目录下的名称为 flower.jpg 的图片，可以将属性值设置为“android:src="@drawable/flower"”
android:tint	用于为图片着色，其属性值可以是“#rgb”、“#argb”、“#rrggbb”或“#aarrggbb”表示的颜色值



说明：

在表 5.6 中，只给出了 ImageView 组件常用的部分属性，关于该组件的其他属性，可以参阅 Android 官方提供的 API 文档。

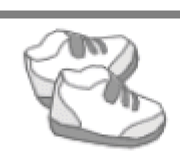
下面将给出一个关于 ImageView 组件的实例。

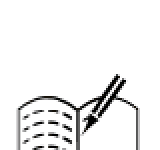
【例 5.17】 在 Eclipse 中创建 Android 项目，主要使用 ImageView 组件显示图像。

实例位置：光盘\MR\Instance\05\5.17

修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在该线性布局管理器中添加一个 ImageView 组件，设置该组件的最大高度和宽度，并且保持纵横比缩放图片，最后为该图像着色，这里设置的是半透明的红色。其代码如下：

```
<ImageView
    android:src="@drawable/image"
```





Note

```
android:id="@+id/imageView2"
android:maxWidth="300px"
android:maxHeight="200px"
android:adjustViewBounds="true"
android:tint="#77ff0000"
android:scaleType="fitEnd"
android:layout_margin="5px"
android:layout_height="wrap_content"
android:layout_width="wrap_content"/>
```

运行本实例，将显示如图 5.18 所示的运行效果。

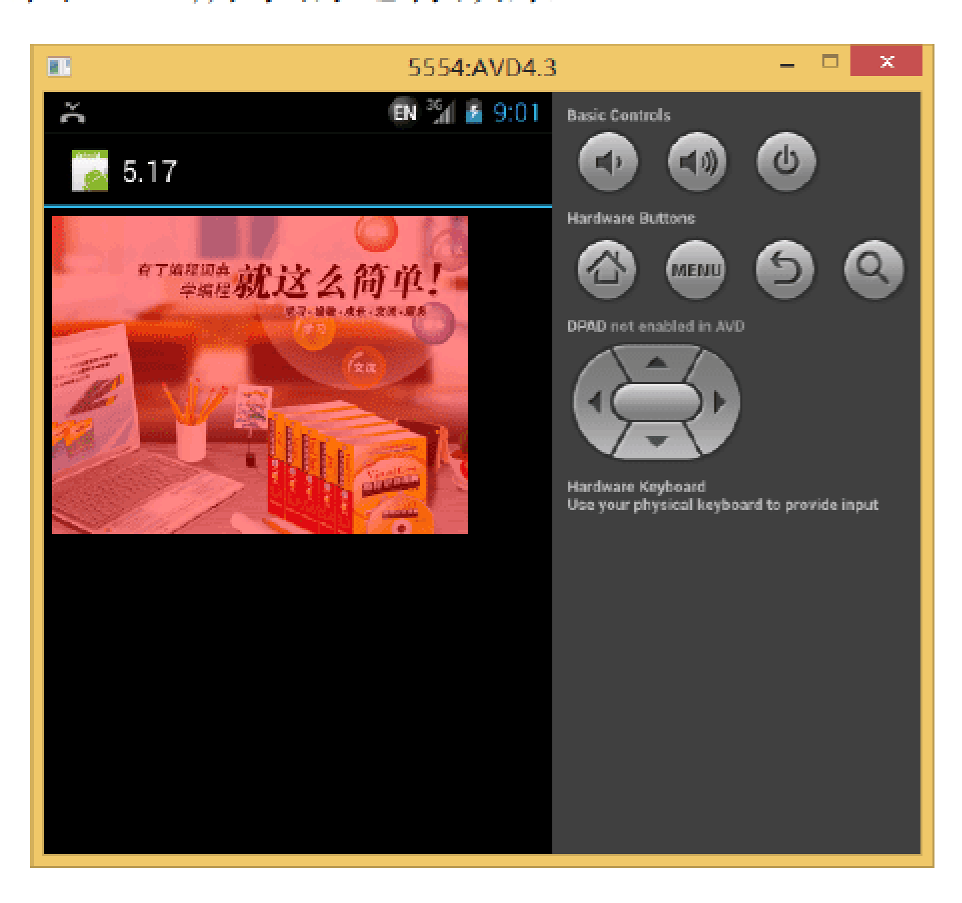


图 5.18　使用 ImageView 显示图像

5.6.2　Gallery 组件

画廊视图使用 Gallery 表示，能够按水平方向显示内容，并且可用手指直接拖动图片移动，一般用来浏览图片，被选中的选项位于中间，并且可以响应事件显示信息。在使用画廊视图时，首先需要在屏幕上添加 Gallery 组件，通常使用<Gallery>标记在 XML 布局文件中添加。在 XML 布局文件中添加画廊视图的基本语法如下：

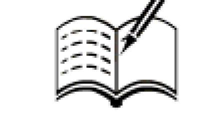
```
< Gallery
    属性列表
/>
```

Gallery 组件支持的 XML 属性如表 5.7 所示。

表 5.7　Gallery 支持的 XML 属性

XML 属性	描　　述
android:animationDuration	用于设置列表项切换时的动画持续时间
android:gravity	用于设置对齐方式
android:spacing	用于设置列表项之间的间距
android:unselectedAlpha	用于设置没有选中的列表项的透明度





使用画廊视图，也需要使用 Adapter 提供要显示的数据，通常使用 BaseAdapter 类为 Gallery 组件提供数据。下面将通过一个具体的实例演示通过 BaseAdapter 适配器为 Gallery 组件提供要显示的图片。

【例 5.18】 在 Eclipse 中创建 Android 项目，主要用来在屏幕中添加画廊视图，实现浏览图片的功能。

实例位置：光盘\MR\Instance\05\5.18

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个 id 属性为 gallery1 的 Gallery 组件，并设置其列表项之间的间距为 5 像素，以及未选中项的透明度。修改后的代码如下：

```
<Gallery
    android:id="@+id/gallery1"
    android:spacing="5px"
    android:unselectedAlpha="0.6"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

（2）在主活动 MainActivity 中，定义一个用于保存要显示图片 id 的数组（需要将要显示的图片复制到 res\drawable 文件夹中）。其关键代码如下：

```
private int[] imageId = new int[] { R.drawable.img01, R.drawable.img02,
        R.drawable.img03, R.drawable.img04, R.drawable.img05 };
```

（3）在主活动的 onCreate()方法中，获取在布局文件中添加的画廊视图。其关键代码如下：

```
Gallery gallery = (Gallery) findViewById(R.id.gallery1);            //获取 Gallery 组件
```

（4）在 res\values 目录的 strings.xml 文件中，定义一个 styleable 对象，用于组合多个属性。这里只指定了一个系统自带的 android:galleryItemBackground 属性，用于设置各选项的背景。其关键代码如下：

```
<resources>
    <declare-styleable name="Gallery">
    <attr name="android:galleryItemBackground" />
    </declare-styleable>
</resources>
```

（5）创建 BaseAdapter 类的对象，并重写其中的 getView()、getItemId()、getItem()和 getCount()方法，其中最主要的是重写 getView()方法来设置显示图片的格式。其具体代码如下：

```
BaseAdapter adapter = new BaseAdapter() {
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageview;                                 //声明 ImageView 的对象
        if (convertView == null) {
```

Note

```
                imageview = new ImageView(MainActivity.this);           //创建 ImageView 的对象
                imageview.setScaleType(ImageView.ScaleType.FIT_XY); //设置缩放方式
                imageview.setLayoutParams(new Gallery.LayoutParams(180, 135));
                TypedArray typedArray = obtainStyledAttributes(R.styleable.Gallery);
                imageview.setBackgroundResource(typedArray.getResourceId(
                        R.styleable.Gallery_android_galleryItemBackground,0));
                imageview.setPadding(5, 0, 5, 0);                    //设置 ImageView 的内边距
            } else {
                imageview = (ImageView) convertView;
            }
            imageview.setImageResource(imageId[position]);           //为 ImageView 设置要显示的图片
            return imageview;                                        //返回 ImageView
        }
        /*
         * 功能：获得当前选项的 id
         */
        @Override
        public long getItemId(int position) {
            return position;
        }
        /*
         * 功能：获得当前选项
         */
        @Override
        public Object getItem(int position) {
            return position;
        }
        /*
         * 获得数量
         */
        @Override
        public int getCount() {
            return imageId.length;
        }
};
```

（6）将步骤（5）中创建的适配器与 Gallery 关联，并且让中间的图片选中，为了在用户单击某张图片时显示对应的位置，还需要为 Gallery 添加单击事件监听。其具体代码如下：

```
gallery.setAdapter(adapter);                                         //将适配器与 Gallery 关联
gallery.setSelection(imageId.length / 2);                            //让中间的图片选中
gallery.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view,int position, long id) {
            Toast.makeText(MainActivity.this,"您选择了第" + String.valueOf(position) + "张图片",
                    Toast.LENGTH_SHORT).show();
        }
});
```

运行本实例，将显示如图 5.19 所示的运行效果。



Note

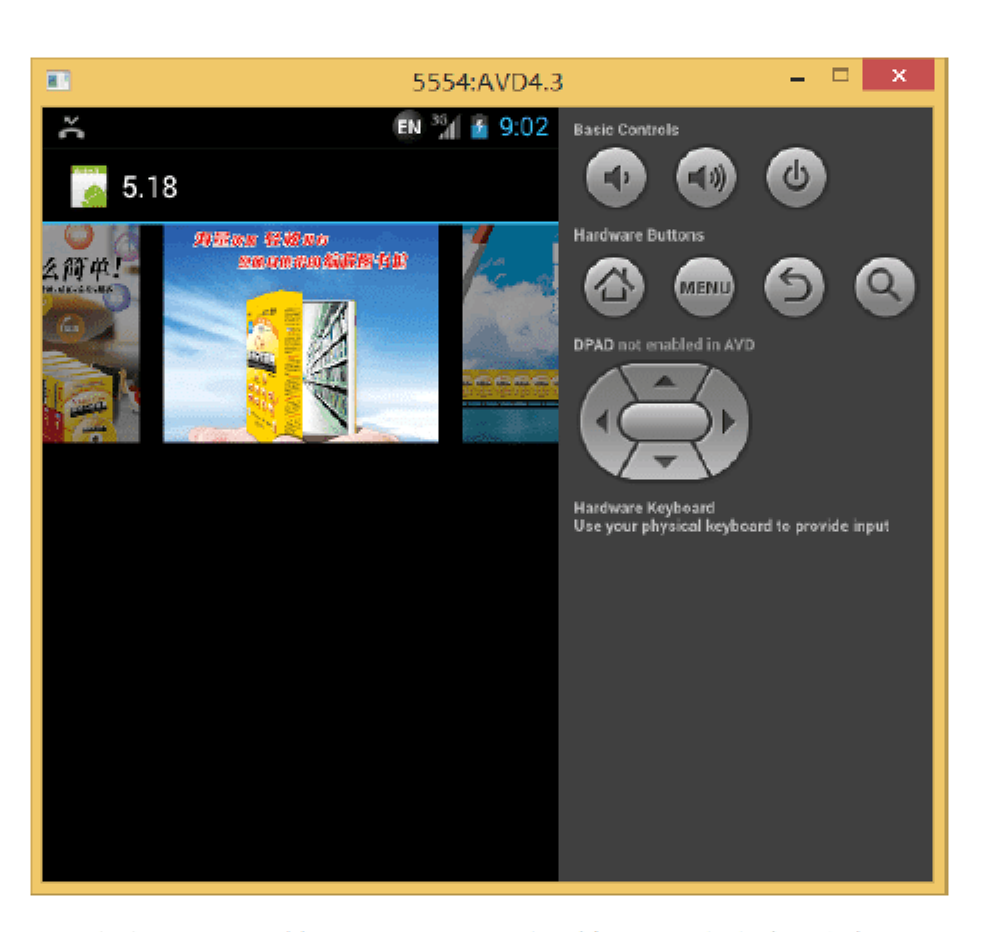




图 5.19　使用 Gallery 组件显示图片列表

5.6.3　ImageSwitcher 组件

图像切换器使用 ImageSwitcher 表示，用于实现类似于 Windows 操作系统下的“Windows 照片查看器”中的上一张、下一张切换图片的功能。在使用 ImageSwitcher 时，必须实现 ViewSwitcher.ViewFactory 接口，并通过 makeView()方法来创建用于显示图片的 ImageView 对象。makeView()方法将返回一个显示图片的 ImageView。在使用 ImageSwitcher 组件时，还有一个方法非常重要，那就是 setImageResource()方法，该方法用于指定要在 ImageSwitcher 中显示的图片资源。下面将通过一个具体的实例来说明 ImageSwitcher 组件的具体用法。

【例 5.19】 在 Eclipse 中创建 Android 项目，使用 ImageSwitcher 组件实现类似 Windows 照片查看器的简单图片查看器。

实例位置：光盘\MR\Instance\05\5.19

程序的开发步骤如下：

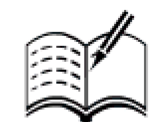
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认的 TextView 组件删除，然后添加两个按钮和一个图像切换器 ImageSwitcher，并设置图像切换器的布局方式为居中显示。其关键代码如下：

```
<Button
    android:text="上一张"
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<ImageSwitcher
    android:id="@+id/imageSwitcher1"
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<Button
    android:text="下一张"
```





```
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```



Note

（2）在主活动 MainActivity 窗口中，首先声明并初始化一个保存要显示图像 id 的数组，然后声明一个保存当前显示图像索引的变量，最后再声明一个图像切换器的对象。其具体代码如下：

```
private int[] imageId = new int[] { R.drawable.img01, R.drawable.img02,R.drawable.img03,
R.drawable.img04, R.drawable.img05 };                    //声明并初始化一个保存要显示图像 id 的数组
private int index = 0;                                   //当前显示图像的索引
private ImageSwitcher imageSwitcher;                     //声明一个图像切换器对象
```

（3）在主活动窗口的 onCreate()方法中，首先获取布局文件中添加的图像切换器，并为其设置淡入淡出的动画效果；然后为其设置一个 ImageSwitcher.ViewFactory，并重写 makeView()方法，在该方法中，设置图像的显示方式、大小等信息；最后为图像切换器设置默认显示的图像。其关键代码如下：

```
imageSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher1);      //获取图像切换器
//设置动画效果
imageSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_in));
//设置淡入动画
imageSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,android.R.anim.fade_out));
//设置淡出动画
imageSwitcher.setFactory(new ViewFactory() {
    @Override
    public View makeView() {
        ImageView imageView = new ImageView(MainActivity.this);//创建 ImageView 类的对象
        imageView.setAdjustViewBounds(true);             //将 AdjustViewBounds 属性设置为 true
        //设置保持纵横比居中缩放图像
        imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
        //设置图像的宽度和高度
        imageView.setLayoutParams(new ImageSwitcher.LayoutParams(240, 180));
        return imageView;                                //返回 imageView 对象
    }
});
imageSwitcher.setImageResource(imageId[index]);          //显示默认的图片
```

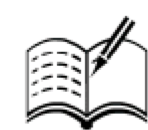
说明：

在上面的代码中，使用 ImageSwitcher 类的父类 ViewAnimator 的 setInAnimation()方法和 setOutAnimation()方法为图像切换器设置动画效果；调用其父类 ViewSwitcher 的 setFactory()方法指定视图切换工厂，其参数为 ViewSwitcher.ViewFactory 类型的对象。

（4）获取用于控制显示图片的“上一张”和“下一张”按钮，并分别为其添加单击事件监听，在重写的 onClick()方法中改变图像切换器中显示的图片。其关键代码如下：

```
Button up = (Button) findViewById(R.id.button1);         //获取“上一张”按钮
Button down = (Button) findViewById(R.id.button2);       //获取“下一张”按钮
up.setOnClickListener(new OnClickListener() {
```





Note

```
        @Override
        public void onClick(View v) {
            if (index > 0) {
                index--;                                            //index 的值-1
            } else {
                index = imageId.length - 1;
            }
            imageSwitcher.setImageResource(imageId[index]);        //显示当前图片
        }
    });
    down.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if (index < imageId.length - 1) {
                index++;                                            //index 的值+1
            } else {
                index = 0;
            }
            imageSwitcher.setImageResource(imageId[index]);        //显示当前图片
        }
    });
```

运行本实例，将显示如图 5.20 所示的运行效果。

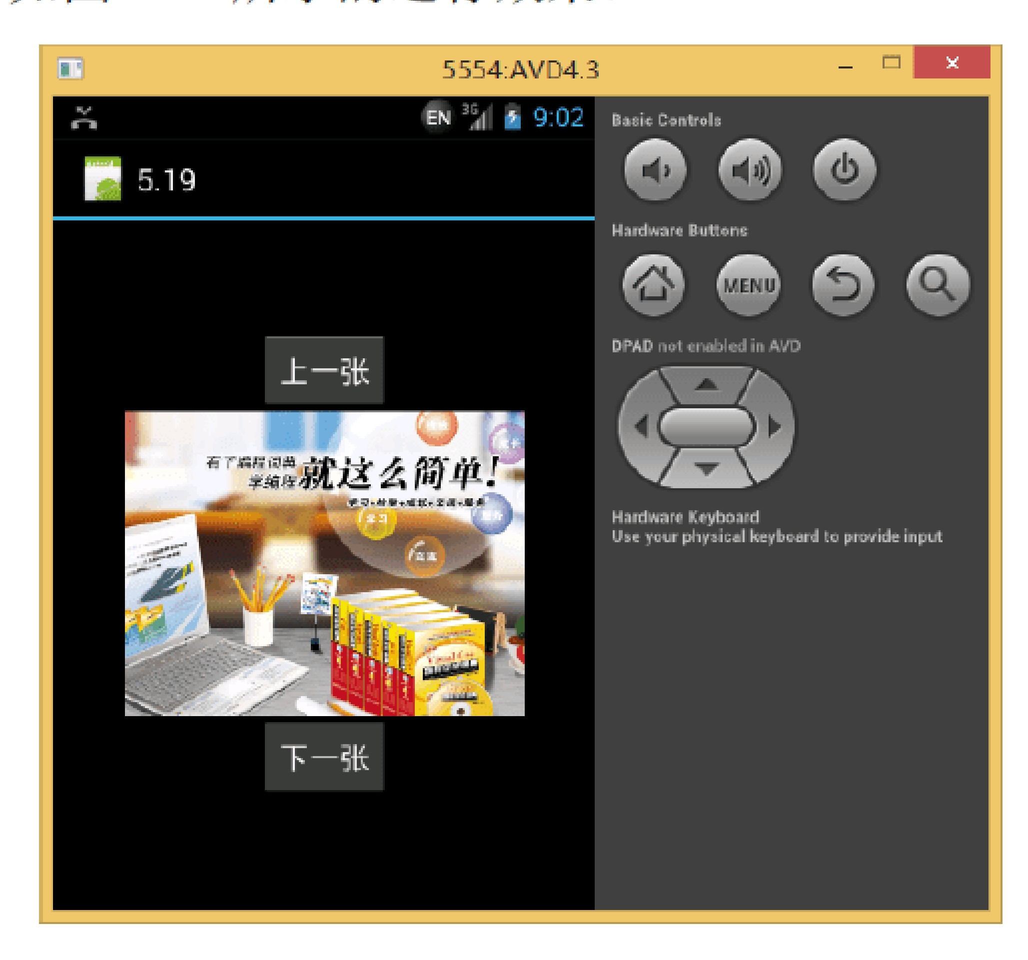


图 5.20　使用 ImageSwitcher 组件实现简单的图片查看器

5.7　综 合 应 用

5.7.1　实现带图标的 ListView 列表

【例 5.20】 在智能手机中，经常会应用到带图标的列表来显示允许操作的功能。本实例将





Note

应用 ListView 组件和 SimpleAdapter 适配器实现一个带图标的 ListView 列表，用于显示手机的常用功能。程序的运行效果如图 5.21 所示。

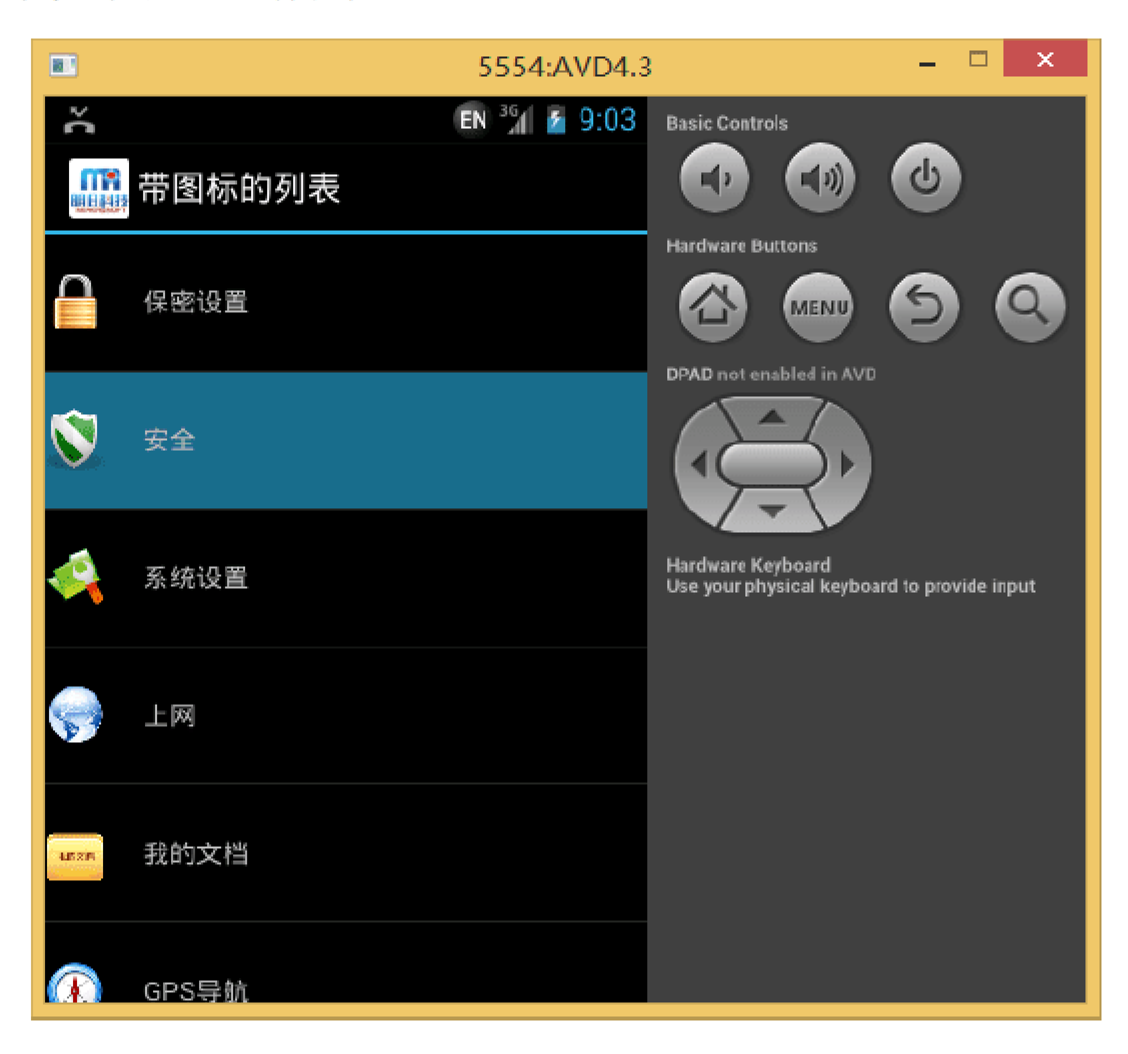


图 5.21　带图标的 ListView

实例位置：光盘\MR\Instance\05\5.20

程序的开发步骤如下：

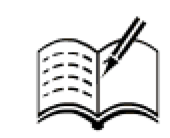
（1）在 Eclipse 中创建 Android 项目，修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个 id 属性为 listView1 的 ListView 组件。修改后的代码如下：

```
<ListView
        android:id="@+id/listView1"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"/>
```

（2）编写用于布局列表项内容的 XML 布局文件 items.xml，在该文件中，采用水平线性布局，并在该布局管理器中添加一个 ImageView 组件和一个 TextView 组件，分别用于显示列表项中的图标和文字。其具体代码如下：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<ImageView
        android:id="@+id/image"
        android:paddingRight="10px"
        android:paddingTop="20px"
        android:paddingBottom="20px"
        android:adjustViewBounds="true"
```






```
        android:maxWidth="72px"
        android:maxHeight="72px"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"/>
  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="10px"
      android:layout_gravity="center"
      android:id="@+id/title"
      />
</LinearLayout>
```

（3）在主活动的 onCreate()方法中，首先获取布局文件中添加的 ListView，然后创建两个用于保存列表项图片 id 和文字的数组，并将这些图片 id 和文字添加到 List 集合中，再创建一个 SimpleAdapter 简单适配器，最后将该适配器与 ListView 相关联。其具体代码如下：

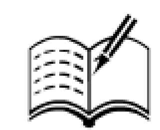
```
ListView listview = (ListView) findViewById(R.id.listView1);//获取列表视图
int[] imageId = new int[] { R.drawable.img01, R.drawable.img02, R.drawable.img03,
                R.drawable.img04, R.drawable.img05, R.drawable.img06,
                R.drawable.img07, R.drawable.img08 };   //定义并初始化保存图片 id 的数组
String[] title = new String[] { "保密设置", "安全", "系统设置", "上网", "我的文档",
        "GPS 导航", "我的音乐", "E-mail" };                //定义并初始化保存列表项文字的数组
//创建 list 集合
List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>();
//通过 for 循环将图片 id 和列表项文字放到 Map 中，并添加到 list 集合中
for (int i = 0; i < imageId.length; i++) {
        Map<String, Object> map = new HashMap<String, Object>();   //实例化 Map 对象
        map.put("image", imageId[i]);
        map.put("title", title[i]);
        listItems.add(map);                                    //将 map 对象添加到 List 集合
}
SimpleAdapter adapter = new SimpleAdapter(this, listItems,
        R.layout.items, new String[] { "title", "image" }, new int[] {
                R.id.title, R.id.image });                     //创建 SimpleAdapter
listview.setAdapter(adapter);                                  //将适配器与 ListView 关联
```



说明：

SimpleAdapter 类的构造方法“SimpleAdapter(Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to)”中，第 1 个参数 context 用于指定关联 SimpleAdapter 运行的视图上下文；第 2 个参数 data 用于指定一个基于 Map 的列表，在该列表中的每个条目对应列表中的一行；第 3 个参数 resource 用于指定一个用于定义列表项目的视图布局文件的唯一标识；第 4 个参数 from 用于指定一个将被添加到 Map 上关联每一个项目的列名称的数组；第 5 个参数 to 用于指定一个与参数 from 显示列对应的视图 id 的数组。

5.7.2 猜猜鸡蛋放在哪只鞋子里


Note

【例 5.21】 使用 Android 开发一个猜猜鸡蛋放在哪只鞋子里的小游戏，运行该程序，将显示如图 5.22 所示的运行效果，单击其中的任意一只鞋子，将打开鞋子显示里面是否有鸡蛋，并且将没有被单击的鞋子设置为半透明显示，被单击的正常显示，同时根据单击的鞋子里面是否有鸡蛋显示对应的结果。例如，单击中间的那只鞋子，如果鸡蛋在这只鞋子里，将显示如图 5.23 所示的运行效果，否则，将显示“很抱歉，猜错了，要不要再试一次？”的提示文字。




图 5.22 默认的运行效果



图 5.23 单击中间的那只鞋子显示的运行效果

 **实例位置：光盘\MR\Instance\05\5.21**

说明：
运行本实例时，需要将 Android 模拟器（AVD）修改为 WSVGA 模式。

程序的开发步骤如下：

（1）在 Eclipse 中创建 Android 项目，修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的垂直线性布局的代码删除，然后添加一个带有 3 个表格行的 TableView 布局，其中第 1 行中，添加一个 TextView 组件，用于显示游戏标题或提示信息；第 2 行添加一个包含 3 个 ImageView 组件的水平线性布局管理器；最后一行是一个水平线性布局管理器，并且在其中添加一个再玩一次按钮。修改后的代码如下：

```
<TableLayout   xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="wrap_content"
    android:background="@drawable/background"
    android:id="@+id/tableLayout1">
    <TableRow android:id="@+id/tableRow1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:gravity="center"
    android:layout_weight="2">
    <TextView
        android:text="@string/title"
        android:padding="10px"
```

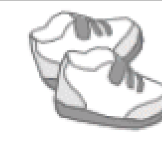


```
            android:gravity="center"
            android:textSize="35px"
            android:textColor="#010D18"
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    </TableRow>
    <TableRow
     android:id="@+id/tableRow2"
     android:layout_weight="1"
     android:gravity="center"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content">
          <LinearLayout
               android:orientation="horizontal"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content" >
               <ImageView android:id="@+id/imageView1"
                android:src="@drawable/shoe_default"
                android:paddingLeft="30px"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"/>
               … <!-- 此处省略其他两个 ImageView 组件的代码，这两个组件一个的 id 属性是
imageView2，另一个是 imageView3 -->
          </LinearLayout>
    </TableRow>
     <LinearLayout
          android:orientation="horizontal"
          android:layout_width="wrap_content" android:layout_height="wrap_content"
          android:layout_weight="1"
          android:gravity="center_horizontal">
         <Button
          android:text="再玩一次"
          android:id="@+id/button1"
          android:layout_width="wrap_content"   android:layout_height="wrap_content"/>
     </LinearLayout>
</TableLayout>
```

（2）在主活动 MainActivity 中，定义 1 个保存全部图片 id 的数组、3 个 ImageView 类型的对象和 1 个 TextView 类型的对象。其具体代码如下：

```
int[] imageIds = new int[] { R.drawable.shoe_ok, R.drawable.shoe_sorry,
                       R.drawable.shoe_sorry };          //定义一个保存全部图片 id 的数组
private ImageView image1;                                //ImageView 组件 1
private ImageView image2;                                //ImageView 组件 2
private ImageView image3;                                //ImageView 组件 3
private TextView result;                                 //显示结果
```

（3）编写一个无返回值的方法 reset()，用于随机指定鸡蛋所在的鞋子。其关键代码如下：

Note

```
private void reset() {
    for (int i = 0; i < 3; i++) {
        int temp = imageIds[i];                    //将数组元素 i 保存到临时变量中
        int index = (int) (Math.random() * 2);  //生成一个随机数
        imageIds[i] = imageIds[index];          //将随机数指定的数组元素的内容赋值给数组元素 i
        imageIds[index] = temp;                  //将临时变量的值赋值给随机数组指定的那个数组元素
    }
}
```

（4）由于 ImageButton 组件设置背景透明后，将不再显示鼠标单击效果，所以需要通过 Drawable 资源来设置图片的 android:src 属性。首先编写一个 Drawable 资源对应的 XML 文件 button_state.xml，用于设置当鼠标按下时显示的图片，以及鼠标没有按下时显示的图片。其具体代码如下：

```
image1 = (ImageView) findViewById(R.id.imageView1);    //获取 ImageView1 组件
image2 = (ImageView) findViewById(R.id.imageView2);    //获取 ImageView2 组件
image3 = (ImageView) findViewById(R.id.imageView3);    //获取 ImageView3 组件
result = (TextView) findViewById(R.id.textView1);       //获取 TextView 组件
reset();                                                  //将鞋子的顺序打乱
```

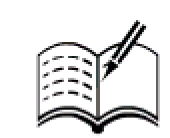
（5）为 3 个显示鞋子的 ImageView 组件添加单击事件监听器，用于将鞋子打开，并显示猜猜看的结果。其关键代码如下：

```
image1.setOnClickListener(new OnClickListener() {      //为第一只鞋子添加单击事件监听
    @Override
    public void onClick(View v) {
        isRight(v, 0);                                  //判断结果
    }
});
image2.setOnClickListener(new OnClickListener() {      //为第二只鞋子添加单击事件监听
    @Override
    public void onClick(View v) {
        isRight(v, 1);                                  //判断结果
    }
});
image3.setOnClickListener(new OnClickListener() {      //为第三只鞋子添加单击事件监听
    @Override
    public void onClick(View v) {
        isRight(v, 2);                                  //判断结果
    }
});
```

（6）编写 isRight()方法，用于显示打开的鞋子，并显示判断结果。其具体代码如下：

```
private void isRight(View v, int index) {
    //使用随机数组中图片资源 id 设置每个 ImageView
    image1.setImageDrawable(getResources().getDrawable(imageIds[0]));
    image2.setImageDrawable(getResources().getDrawable(imageIds[1]));
    image3.setImageDrawable(getResources().getDrawable(imageIds[2]));
```






```
        //为每个 ImageView 设置半透明效果
        image1.setAlpha(100);
        image2.setAlpha(100);
        image3.setAlpha(100);
        ImageView v1 = (ImageView) v;                         //获取被单击的图像视图
        v1.setAlpha(255);                                     //设置图像视图的透明度
        if (imageIds[index] == R.drawable.shoe_ok) {          //判断是否猜对
            result.setText("恭喜您，猜对了，祝你幸福！");
        } else {
            result.setText("很抱歉，猜错了，要不要再试一次？");
        }
}
```

（7）获取“再玩一次”按钮，并为该按钮添加单击事件监听器，在其单击事件中，首先将标题恢复为默认值，然后设置 3 个 ImageView 的透明度为完全不透明，最后再设置这 3 个 ImageView 的图像内容为默认显示图片。其具体代码如下：

```
Button button = (Button) findViewById(R.id.button1);     //获取“再玩一次”按钮
button.setOnClickListener(new OnClickListener() {        //为“再玩一次”按钮添加事件监听器
        @Override
        public void onClick(View v) {
            reset();
            result.setText(R.string.title);              //将标题恢复为默认值
            image1.setAlpha(255);
            image2.setAlpha(255);
            image3.setAlpha(255);
            image1.setImageDrawable(getResources().getDrawable( R.drawable.shoe_default));
            image2.setImageDrawable(getResources().getDrawable(R.drawable.shoe_default));
            image3.setImageDrawable(getResources().getDrawable(R.drawable.shoe_default));
        }
});
```

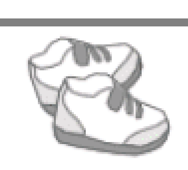
5.8　本章常见错误

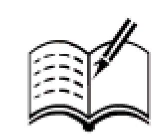
开发 Android 程序时，想通过 ScrollView 实现 ListView 的滚动效果，所以就在 ScrollView 组件中嵌套了一个 ListView 组件，并且将 layout_height 属性设置为了 fill_partent，但在运行时，发现无论 ListView 中添加多少行数据，在界面上都只能显示两行。

经过分析发现，当子控件的高度小于 ScrollView 的高度时，如果想让这个子控件 fill_partent 显示，单独定义“android:layout_height="fill_parent"”是不起作用的，还需要为 ScrollView 组件加上 fillViewport 属性；而如果子控件的高度大于 ScrollView 的高度时，就无须设置 fillViewport 属性了。因此，要避免上面所描述的错误，可以有以下两种方法。

☑　为 ScrollView 组件添加“android:fillViewport="true"”属性。

☑　将 ListView 组件的高度设置大一些，如“android:layout_height="500dp"”。





Note

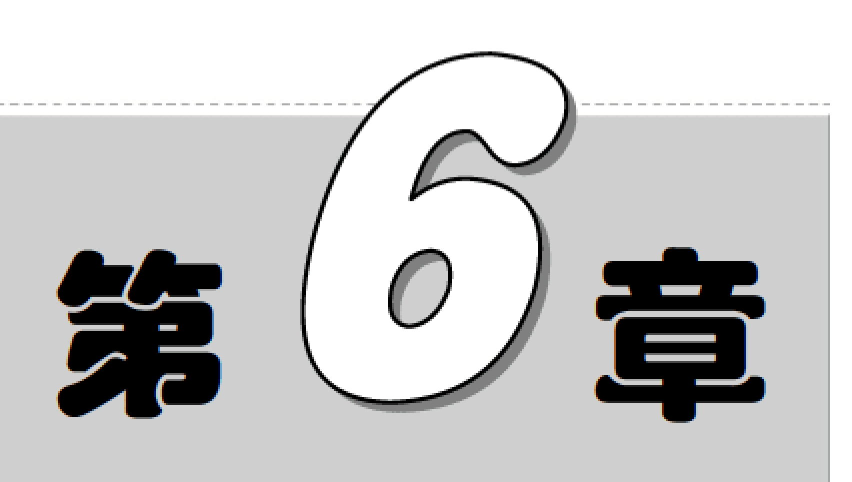
5.9 本章小结

本章首先对 Android 程序的 UI 界面设计进行了详细讲解，其中主要讲解了 4 种 UI 界面设计方式，读者需要重点掌握如何使用 XML 布局文件控制 UI 界面；然后重点讲解了 Android 程序开发时经常用到的 5 类组件及其使用，分别是文本类组件、按钮类组件、选择类组件、列表类组件和图像类组件，这些组件是开发 Android 程序时最基本、同时也是最重要的内容，所以在学习本章内容时，一定要熟练掌握本章所讲解的各类组件，并能够将它们应用到实际开发中。

5.10 跟我上机

参考答案：光盘\MR\跟我上机

开发一个 Android 程序，要求使用本章讲解的 ImageButton 组件制作一个跟踪鼠标单击状态的图片按钮。具体实现时，首先需要修改新建项目的 res\layout 目录下的布局文件 main.xml，为默认添加的垂直线性布局添加背景，设置该布局中的内容居中显示，同时，添加一个 ImageButton 图片按钮，将其设置为透明背景；然后编写 Drawable 资源对应的 XML 文件 button_state.xml，用于设置当鼠标按下时显示的图片（start_a.png），以及鼠标没有按下时显示的图片（start_b.png），编写完该文件之后，为 main.xml 布局文件中的图片按钮设置 android:src 属性，其属性值是刚才所编写的 Drawable 资源；最后，在主活动 Activity 的 onCreate()方法中，获取布局文件中添加的图片按钮，并为其添加鼠标单击事件监听器，使用 Toast.makeText()方法弹出“进入游戏...”的信息提示。



掌握布局管理器

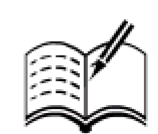
（视频讲解：58 分钟）

在 Android 程序中，每个组件在容器中都有一个具体的位置和大小。在容器中摆放各种组件时，很难判断其具体位置和大小，而布局管理器提供了在 Android 程序中安排展示组件的方法。通过使用布局管理器，开发人员可以很方便地在容器中控制组件的位置和大小，以便有效地管理整个窗体的布局。Android 中主要提供了线性布局、绝对布局、框架布局、相对布局和表格布局 5 种管理器，本章将分别对它们进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 使用线性布局管理器布局 Android 界面
- ☐ 使用绝对布局固定组件的位置
- ☐ 使用框架布局居中显示层叠的正方形
- ☐ 使用相对布局管理器布局多个组件的相对位置
- ☐ 使用表格布局管理器布局用户的登录界面
- ☐ 我同意游戏条款
- ☐ 应用相对布局显示软件更新提示
- ☐ 布局个性游戏开始界面

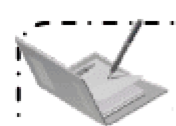




Note

6.1 线性布局管理器

Android 中的线性布局管理器用 LinearLayout 表示，它是将放入其中的组件按照垂直或水平方向来布局，也就是控制放入其中的组件横向排列或纵向排列。在线性布局中，每一行（针对垂直排列）或每一列（针对水平排列）中只能放一个组件，并且 Android 的线性布局不会换行，当组件一个挨着一个排列到窗体的边缘后，剩下的组件将不会被显示出来。



说明：

在线性布局中，排列方式由 android:orientation 属性来控制，对齐方式由 android:gravity 属性来控制。

在 Android 中，可以在 XML 布局文件中定义线性布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义线性布局管理器。在 XML 布局文件中定义线性布局管理器，需要使用<LinearLayout>标记。其基本语法格式如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    属性列表
>
</LinearLayout>
```

在线性布局管理器中，常用的属性包括 android:orientation、android:gravity、android:layout_width、android:layout_height、android:id 和 android:background。其中，前两个属性是线性布局管理器支持的属性，后面的 4 个是 android.view.View 和 android.view.ViewGroup 支持的属性，下面进行详细介绍。

☑ android:orientation 属性

android:orientation 属性用于设置布局管理器内组件的排列方式，其可选值包括 horizontal 和 vertical，默认值为 vertical。其中，horizontal 表示水平排列；vertical 表示垂直排列。

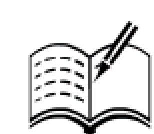
☑ android:gravity 属性

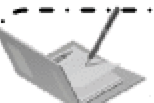
android:gravity 属性用于设置布局管理器内组件的对齐方式，其可选值包括 top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 和 clip_horizontal。这些属性值也可以同时指定，各属性值之间用竖线隔开。例如要指定组件靠右下角对齐，可以使用属性值 right|bottom。

☑ android:layout_width 属性

android:layout_width 属性用于设置布局管理器内组件的基本宽度，其可选值包括 fill_parent、match_parent 和 wrap_content。其中，fill_parent 表示该组件的宽度与父容器的宽度相同；match_paren 与 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示该组件的宽度恰好能包裹它的内容即可。





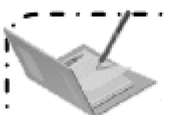


说明：

android:layout_width 属性是 ViewGroup.LayoutParams 所支持的 XML 属性，对于其他布局管理器同样适用。

☑ android:layout_height 属性

android:layout_width 属性用于设置布局管理器内组件的基本高度，其可选值包括 fill_parent、match_parent 和 wrap_content。其中，fill_parent 表示该组件的高度与父容器的高度相同；match_parent 与 fill_parent 的作用完全相同，从 Android 2.2 开始推荐使用；wrap_content 表示该组件的高度恰好能包裹它的内容即可。



说明：

android:layout_height 属性是 ViewGroup.LayoutParams 所支持的 XML 属性，对于其他布局管理器同样适用。

☑ android:id 属性

android:id 属性用于为当前组件指定一个 id 属性，在 Java 代码中可以应用该属性单独引用这个组件。为组件指定 id 属性后，在 R.java 文件中，会自动派生一个对应的属性。在 Java 代码中，可以通过 findViewById()方法来获取它。

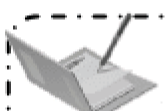
☑ android:background 属性

android:background 属性用于为该组件设置背景。可以是背景图片，也可以是背景颜色。为组件指定背景图片时，可以将准备好的背景图片复制到目录下，然后使用下面的代码进行设置。

```
android:background="@drawable/background"
```

如果想指定背景颜色，可以使用颜色值。例如，要想指定背景颜色为白色，可以使用下面的代码。

```
android:background="#FFFFFFFF"
```



说明：

在线性布局中，还可以使用 android.view.View 类支持的其他属性，更加详细的内容可以参阅 Android 官方提供的 API 文档。

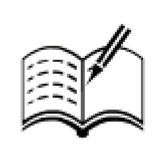
下面通过一个实例讲解如何在 Android 程序中使用线性布局管理器。

【例 6.1】 在 Eclipse 中创建 Android 项目，实现采用线性布局管理器布局 Android 界面。

实例位置：光盘\MR\Instance\06\6.1

修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认添加的垂直线性布局管理器<LinearLayout>中添加两个嵌套的<LinearLayout>，然后设置第一个<LinearLayout>的排列方式为水平排列，在其中添加 4 个水平并排的 TextView 组件，并分别设置 TextView 组件的文本对齐方式；设置第二个<LinearLayout>的排列方式为垂直排列，并在其中添加 4 个垂直并排的 TextView





Note

组件。修改后的代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <LinearLayout
     android:orientation="horizontal"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:layout_weight="1">
     <TextView
         android:layout_width="wrap_content"
         android:layout_height="fill_parent"
         android:text="红色"
         android:gravity="center"
         android:background="#aa0000"
         android:layout_weight="1"
     />
     <TextView
         android:layout_width="wrap_content"
         android:layout_height="fill_parent"
         android:text="蓝色"
         android:gravity="top|center"
         android:background="#0000aa"
         android:layout_weight="1"
     />
     <TextView
         android:layout_width="wrap_content"
         android:layout_height="fill_parent"
         android:text="黄色"
         android:gravity="bottom|center"
         android:background="#aaaa00"
         android:layout_weight="1"
     />
     <TextView
         android:layout_width="wrap_content"
         android:layout_height="fill_parent"
         android:text="绿色"
         android:gravity="fill_vertical"
         android:background="#00aa00"
         android:layout_weight="1"
     />
    </LinearLayout>
     <LinearLayout
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:layout_weight="1">
```





Note

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="第一行"
        android:layout_weight="1"
    />
       <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="第二行"
        android:layout_weight="1"
    />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="第三行"
        android:layout_weight="1"
    />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="第四行"
        android:layout_weight="1"
    />
  </LinearLayout>
</LinearLayout>
```

说明：

上面的代码中用到了 android:layout_weight 属性，该属性用来设置组件的占用空间，如在线性布局中添加 3 个 TextView 组件，它们的 android:layout_weight 属性分别设置为 2、1、1，则它们所占用的空间分别为 1/2、1/4 和 1/4。

运行本实例，将显示如图 6.1 所示的运行效果。

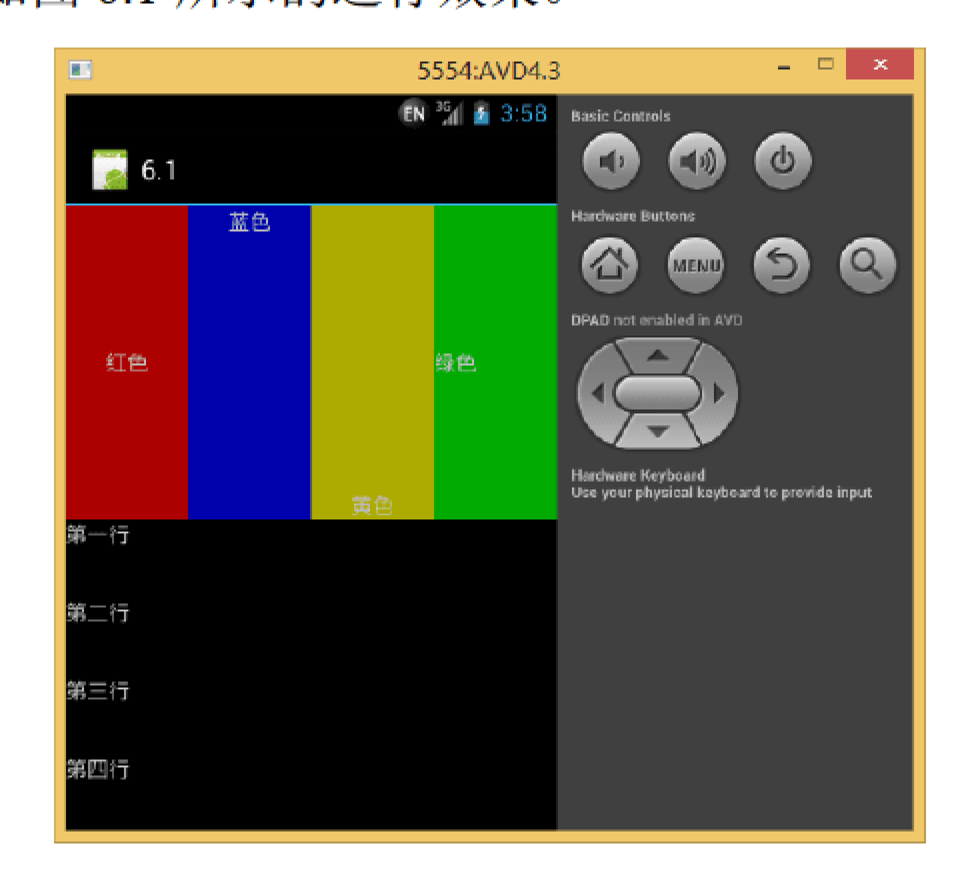
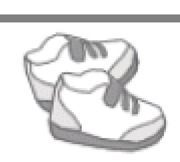
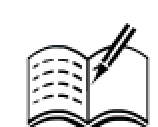


图 6.1　使用线性布局管理器布局 Android 界面





Note

6.2　绝对布局管理器

绝对布局管理器用<AbsoluteLayout>表示，在使用该布局方式时，需要指定子控件的 xy 精确坐标。在 Android 中，可以在 XML 布局文件中定义绝对布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义绝对布局管理器。在 XML 布局文件中，定义绝对布局管理器可以使用<AbsoluteLayout>标记。其基本语法格式如下：

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
>
</AbsoluteLayout>
```

下面通过一个实例讲解如何在 Android 程序中使用绝对布局管理器。

【例 6.2】 在 Eclipse 中创建 Android 项目，使用绝对布局管理器控制一个 TextView 组件和一个 Button 组件的绝对位置。

实例位置：光盘\MR\Instance\06\6.2

修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个绝对布局管理器<AbsoluteLayout>，然后分别添加一个 TextView 组件和一个 Button 组件，并分别通过 android:layout_x 和 android:layout_y 属性设置它们的绝对位置。修改后的代码如下：

```
<AbsoluteLayout  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
     <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_x="20px"
        android:layout_y="10px"
        android:text="用户名："
        />
     <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_x="50px"
        android:layout_y="100px"
        android:text="确定"
        />
</AbsoluteLayout>
```

运行本实例，将显示如图 6.2 所示的运行效果。





Note

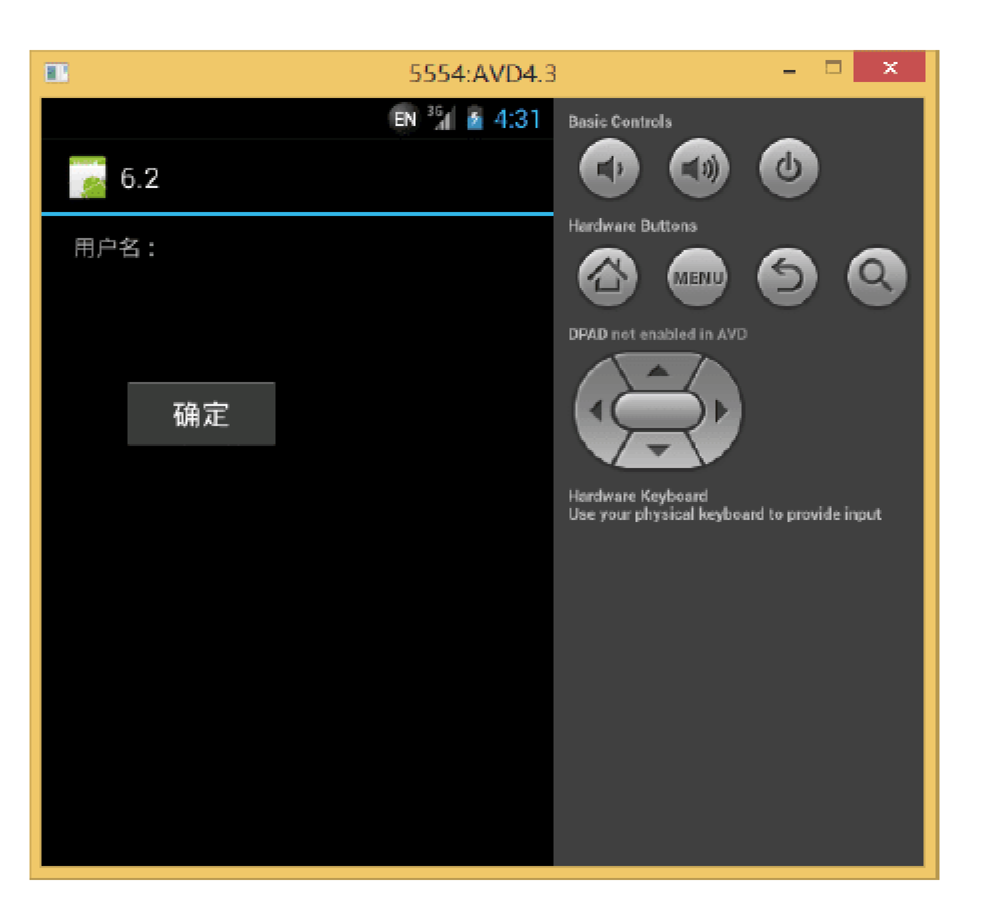


图 6.2　使用绝对布局管理器固定组件的位置



说明：

绝对布局管理器缺乏灵活性，在没有绝对定位的情况下相比其他类型的布局更难维护，所以，在程序中一般不推荐使用这种方式。

6.3　框架布局管理器

框架布局管理器用<FrameLayout>表示，在该布局管理器中，每加入一个组件，都将创建一个空白的区域，通常称为一帧，这些帧都会根据 gravity 属性执行自动对齐。默认情况下，框架布局是从屏幕的左上角（0,0）坐标点开始布局，多个组件层叠排序，后面的组件覆盖前面的组件。

说明：

框架布局管理器，也被称为帧布局管理器。

在 Android 中，可以在 XML 布局文件中定义框架布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义框架布局管理器。在 XML 布局文件中，定义框架布局管理器可以使用<FrameLayout>标记。其基本语法格式如下：

```
< FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
>
</ FrameLayout>
```

FrameLayout 支持的常用 XML 属性如表 6.1 所示。

表 6.1　FrameLayout 支持的常用 XML 属性

XML 属性	描　　述
android:foreground	设置该框架布局容器的前景图像
android:foregroundGravity	定义绘制前景图像的 gravity 属性，也就是前景图像显示的位置



下面通过一个实例讲解如何在 Android 程序中使用框架布局管理器。

【例 6.3】 在 Eclipse 中创建 Android 项目，应用框架布局管理器居中显示层叠的正方形。

实例位置：光盘\MR\Instance\06\6.3

Note

修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个框架布局管理器<FrameLayout>，最后在该布局管理器中添加 3 个居中显示的 TextView，并且分别为它们指定不同的颜色和大小，用于更好地体现层叠效果。修改后的代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
    android:layout_width="280dp"
    android:layout_height="280dp"
    android:background="#004433"
    android:layout_gravity="center"
    />
    <TextView
    android:layout_width="240dp"
    android:layout_height="240dp"
    android:background="#00aa00"
    android:layout_gravity="center"
    />
    <TextView
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:background="#00dd00"
    android:layout_gravity="center"
    />
</FrameLayout>
```

运行本实例，将显示如图 6.3 所示的运行效果。

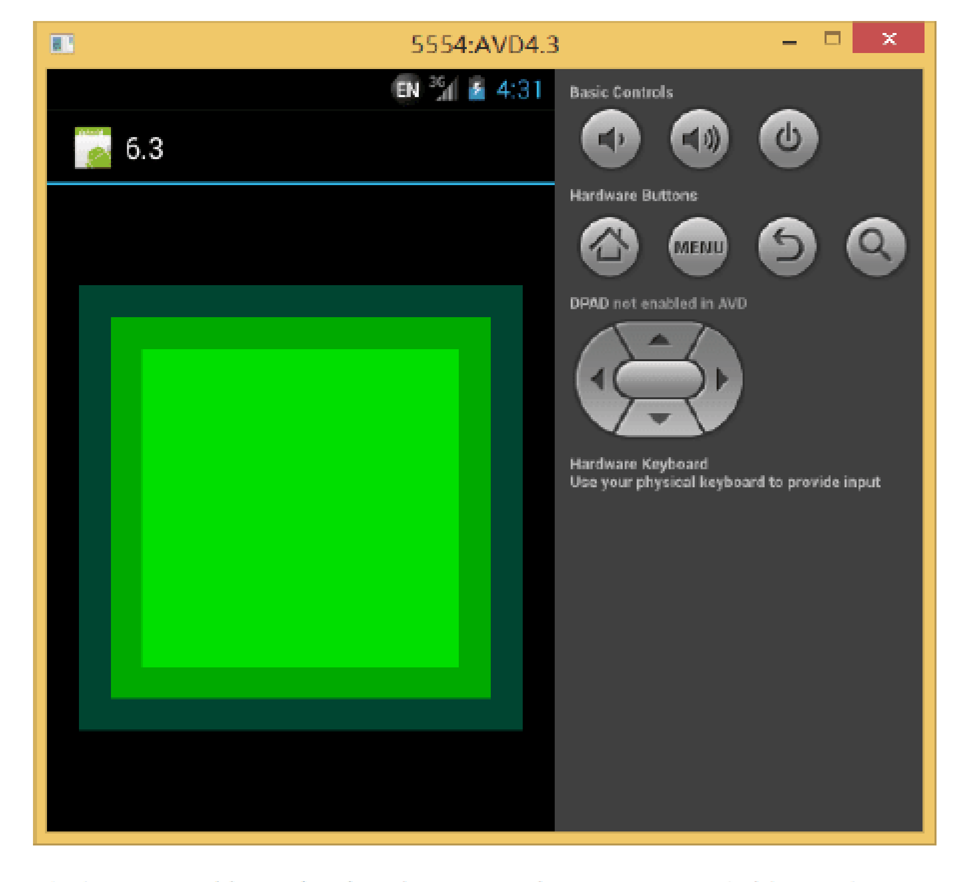
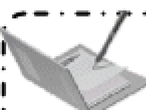


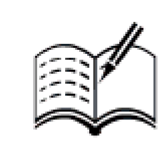
图 6.3　使用框架布局居中显示层叠的正方形





说明：

框架布局管理器经常应用在游戏开发中，用于显示自定义的视图。



6.4　相对布局管理器

Android 中的相对布局管理器用<RelativeLayout>来表示，它是指按照组件之间的相对位置来进行布局，如某个组件在另一个组件的左边、右边、上面或下面等。

在 Android 中，可以在 XML 布局文件中定义相对布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义相对布局管理器。在 XML 布局文件中，定义相对布局管理器可以使用<RelativeLayout>标记。其基本语法格式如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
>
</RelativeLayout>
```

RelativeLayout 支持的常用 XML 属性如表 6.2 所示。

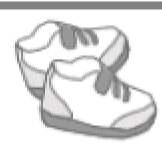
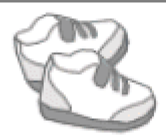
表 6.2　RelativeLayout 支持的常用 XML 属性

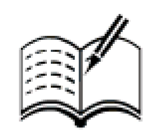
XML 属性	描　　述
android:gravity	用于设置布局管理器中各子组件的对齐方式
android:ignoreGravity	用于指定哪个组件不受 gravity 属性的影响

在相对布局管理器中，只有表 6.2 介绍的两个属性是不够的，为了更好地控制该布局管理器中各子组件的布局分布，RelativeLayout 提供了一个内部类 RelativeLayout.LayoutParams，通过该类提供的大量 XML 属性可以很好地控制相对布局管理器中各组件的分布方式。RelativeLayout.LayoutParams 提供的 XML 属性如表 6.3 所示。

表 6.3　RelativeLayout.LayoutParams 提供的 XML 属性

XML 属性	描　　述
android:layout_above	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的上方
android:layout_alignBottom	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的下边界对齐
android:layout_alignLeft	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的左边界对齐
android:layout_alignParentBottom	其属性值为 boolean 值，用于指定该组件是否与布局管理器底端对齐
android:layout_alignParentLeft	其属性值为 boolean 值，用于指定该组件是否与布局管理器左边对齐
android:layout_alignParentRight	其属性值为 boolean 值，用于指定该组件是否与布局管理器右边对齐
android:layout_alignParentTop	其属性值为 boolean 值，用于指定该组件是否与布局管理器顶端对齐
android:layout_alignRight	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的右边界对齐
android:layout_alignTop	其属性值为其他 UI 组件的 id 属性，用于指定该组件与哪个组件的上边界对齐
android:layout_below	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的下方



续表

XML 属性	描　述
android:layout_centerHorizontal	其属性值为 boolean 值，用于指定该组件是否位于布局管理器水平居中的位置
android:layout_centerInParent	其属性值为 boolean 值，用于指定该组件是否位于布局管理器的中央位置
android:layout_centerVertical	其属性值为 boolean 值，用于指定该组件是否位于布局管理器垂直居中的位置
android:layout_toLeftOf	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的左侧
android:layout_toRightOf	其属性值为其他 UI 组件的 id 属性，用于指定该组件位于哪个组件的右侧

下面通过一个实例讲解如何在 Android 程序中使用相对布局管理器。

【例 6.4】 在 Eclipse 中创建 Android 项目，使用相对布局管理器布局“手机号码”文本框、“确定”按钮和“取消”按钮的相对位置。

实例位置：光盘\MR\Instance\06\6.4

修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个相对布局管理器<RelativeLayout>，在该布局管理器中添加一个 TextView、一个 EditText 组件和两个 Button 组件，并设置它们的显示位置及对齐方式。修改后的代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    >
<TextView
android:id="@+id/txt"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="手机号码："
    android:layout_marginBottom="5dp"
    />
    <EditText
    android:id="@+id/edit"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/txt"
    android:inputType="number"
    android:numeric="integer"
    android:maxLength="11"
    android:hint="请输入手机号码"
    />
    <Button
    android:id="@+id/btn1"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/edit"
    android:layout_alignParentRight="true"
    android:layout_marginLeft="10dp"
    android:text="取消"
```






```
    />
    <Button
     android:id="@+id/btn2"
     android:layout_width="80dp"
     android:layout_height="wrap_content"
     android:layout_below="@id/edit"
     android:layout_toLeftOf="@id/btn1"
     android:text="确定"
    />
</RelativeLayout>
```

注意：

在上面的代码中，首先设置按钮 edit 在 txt 的下方显示，然后设置 btn1 在 edit 的下方居右显示，最后设置按钮 btn2 在 edit 的下方、btn1 的左侧显示。

运行本实例，将显示如图 6.4 所示的运行效果。

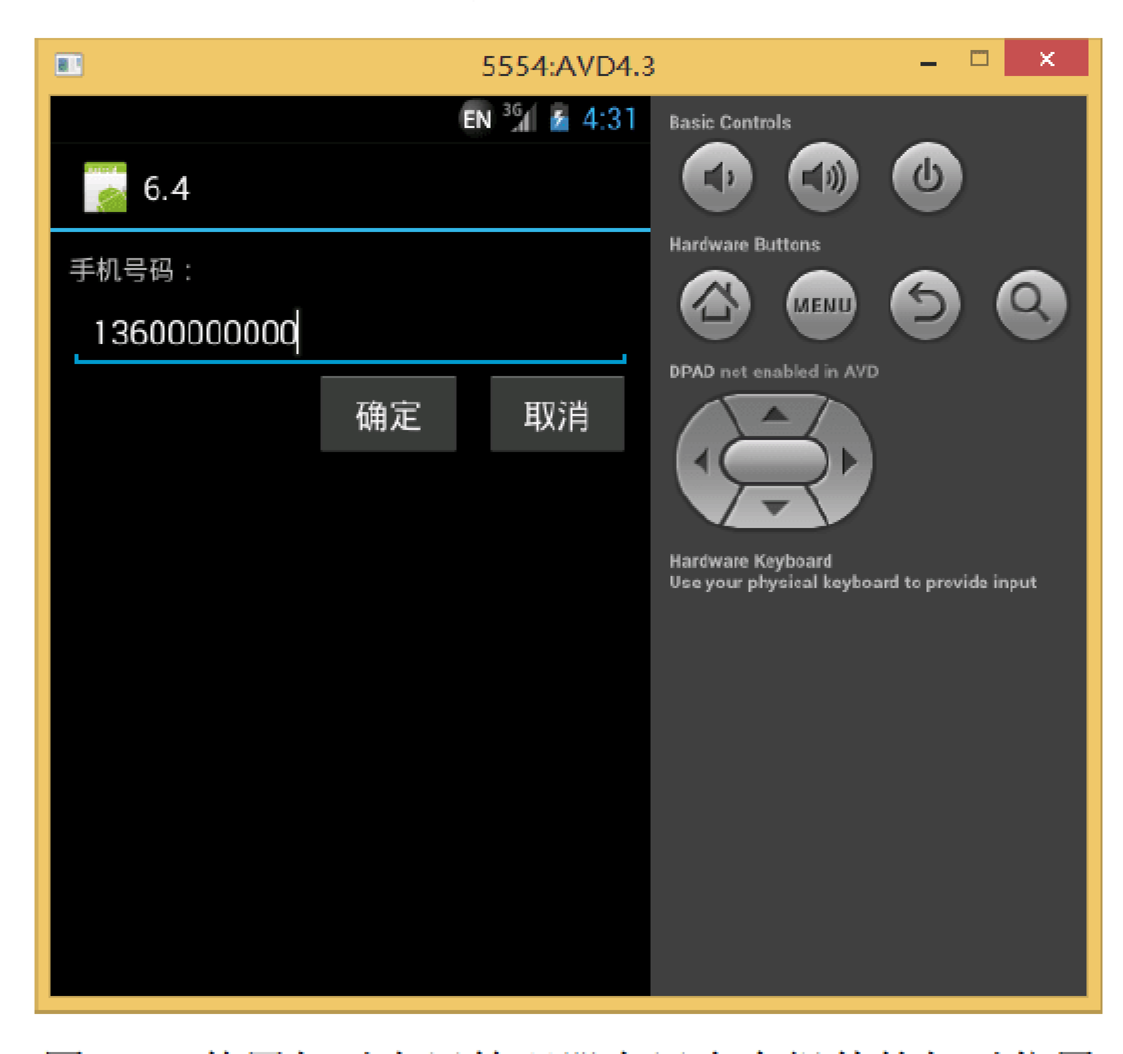
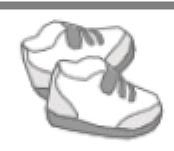


图 6.4　使用相对布局管理器布局多个组件的相对位置

6.5　表格布局管理器

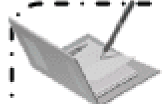
表格布局管理器与常见的表格类似，它以行、列的形式来管理放入其中的 UI 组件。表格布局管理器使用<TableLayout>标记定义。在表格布局管理器中，可以添加多个<TableRow>标记，每个<TableRow>标记占用一行，由于<TableRow>标记也是容器，所以在该标记中还可添加其他组件。在<TableRow>标记中，每添加一个组件，表格就会增加一列。在表格布局管理器中，列可以被隐藏，也可以被设置为伸展的，从而填充可利用的屏幕空间，也可以设置为强制收缩，直到表格匹配屏幕大小。





说明：

如果在表格布局管理器中直接向<TableLayout>标记中添加 UI 组件，那么这个组件将独占一行。



Note

在 Android 中，可以在 XML 布局文件中定义表格布局管理器，也可以使用 Java 代码来创建。推荐使用在 XML 布局文件中定义表格布局管理器。其基本语法格式如下：

```
<TableLayout   xmlns:android="http://schemas.android.com/apk/res/android"
属性列表
>
    <TableRow 属性列表> 需要添加的 UI 组件 </TableRow>
    <!-- 多个<TableRow> -->
</TableLayout>
```

TableLayout 继承了 LinearLayout，因此它完全支持 LinearLayout 所支持的全部 XML 属性，此外，TableLayout 还支持如表 6.4 所示的 XML 属性。

表 6.4　TableLayout 支持的 XML 属性

XML 属性	描　述
android:collapseColumns	设置需要被隐藏的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔
android:shrinkColumns	设置允许被收缩的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔
android:stretchColumns	设置允许被拉伸的列的列序号（序号从 0 开始），多个列序号之间用逗号“,”分隔

下面通过一个实例讲解如何在 Android 程序中使用表格布局管理器。

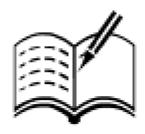
【例 6.5】 在 Eclipse 中创建 Android 项目，使用表格布局管理器布局用户的登录界面。

实例位置：光盘\MR\Instance\06\6.5

修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个表格布局管理器<TableLayout>，并且在该布局管理器中添加 3 个<TableRow>表格行，接下来再在每个表格行中添加用户登录界面相关的组件，最后设置表格的第 1 列和第 4 列允许被拉伸。修改后的代码如下：

```
<TableLayout android:id="@+id/tableLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:gravity="center_vertical"
    android:stretchColumns="0,3"
    >
    <!-- 第 1 行 -->
    <TableRow android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView/>
        <TextView android:text="用户名："
```





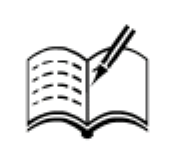

```
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:textSize="24px"
                android:layout_height="wrap_content"
                />
            <EditText android:id="@+id/editText1"
                android:textSize="24px"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" android:minWidth="200px"/>
            <TextView />
        </TableRow>
        <!-- 第 2 行 -->
        <TableRow android:id="@+id/tableRow2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <TextView/>
            <TextView android:text="密    码："
                android:id="@+id/textView2"
                android:textSize="24px"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
            <EditText android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:textSize="24px"
                android:id="@+id/editText2"
                android:inputType="textPassword"/>
            <TextView />
        </TableRow>
        <!-- 第 3 行 -->
        <TableRow android:id="@+id/tableRow3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <TextView/>
            <Button android:text="登录"
                android:id="@+id/button1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
            <Button android:text="退出"
                android:id="@+id/button2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
            <TextView />
        </TableRow>
</TableLayout>
```



说明：

在本实例中，添加了 6 个<TextView />组件，并且设置对应列允许拉伸，是为了让用户登录表单在水平方向上居中显示而设置的。





Note

运行本实例，将显示如图 6.5 所示的运行效果。

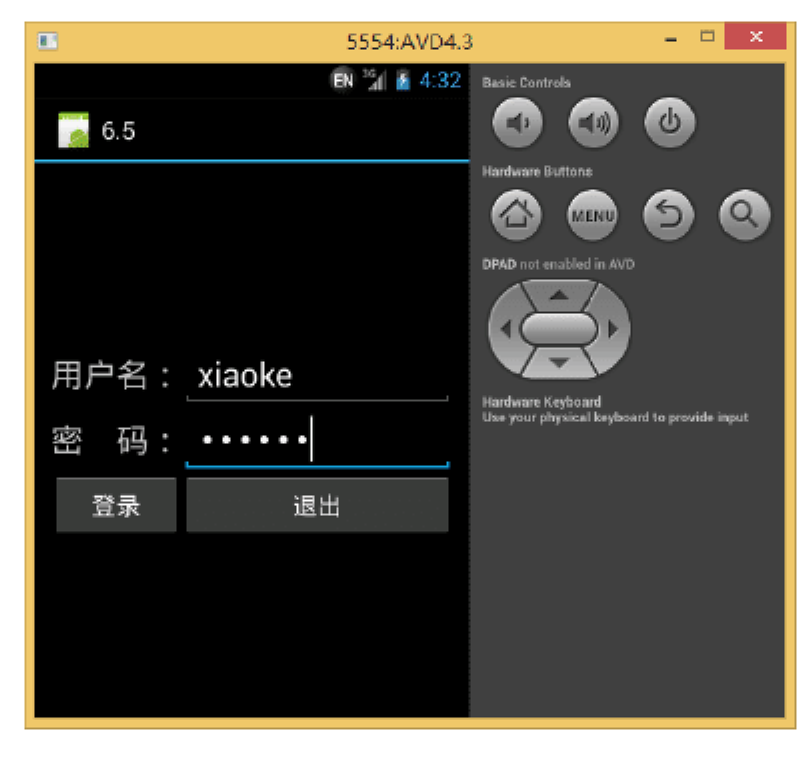


图 6.5　使用表格布局管理器布局用户的登录界面

6.6　综 合 应 用

6.6.1　我同意游戏条款

【例 6.6】本实例要求使用 Android 中的布局管理器实现一个游戏开始界面中的我同意游戏条款功能，运行程序，将显示如图 6.6 所示的运行效果。

实例位置：光盘\MR\Instance\06\6.6

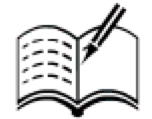


图 6.6　我同意游戏条款的效果

程序的开发步骤如下：

（1）本实例实现时，需要使用垂直线性布局管理器，并且需要借助 Android 中的 TextView、CheckBox 和 ImageButton 组件，其中，TextView 组件用于显示游戏条款；CheckBox 组件用来作为“我同意”复选框；ImageButton 组件用来作为“进入”图片按钮，需要设置图片按钮默认为不显示，以及透明背景。界面布局的代码如下：





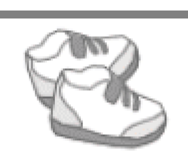
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
        android:gravity="center"
    >
    <TextView
     android:text="@string/artcle"
     android:id="@+id/textView1"
     android:paddingTop="120px"
     style="@style/artclestyle"
     android:maxWidth="700px"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
    <CheckBox
     android:text="我同意"
     android:id="@+id/checkBox1"
     android:textSize="22px"
     android:button="@drawable/check_box"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"/>
     <ImageButton
          android:id="@+id/start"
          android:src="@drawable/button_state"
          android:background="#0000"
          android:paddingTop="30px"
          android:visibility="invisible"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content">
     </ImageButton>
</LinearLayout>
```

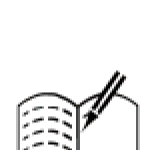
（2）由于复选框默认的效果显示到本实例的绿色背景上时看不到前面的方块，所以需要改变复选框的默认效果。首先编写 Drawable 资源对应的 XML 文件 check_box.xml，用于设置复选框没有被选中时显示的图片，以及被选中时显示的图片。其具体代码如下：

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_checked="false"
          android:drawable="@drawable/check_f"/>
    <item android:state_checked="true"
          android:drawable="@drawable/check_t"/>
</selector>
```

（3）为 main.xml 布局文件中的复选框设置 android:button 属性，其属性值是在步骤（2）中编写的 Drawable 资源。其关键代码如下：

```
android:button="@drawable/check_box"
```





（4）由于 ImageButton 组件设置背景透明后，将不再显示鼠标单击效果，所以需要通过 Drawable 资源来设置图片的 android:src 属性。首先编写一个 Drawable 资源对应的 XML 文件 button_state.xml，用于设置当鼠标按下时显示的图片，以及鼠标没有按下时显示的图片。其具体代码如下：

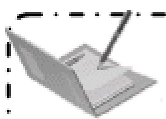
```
<?xml version="1.0" encoding="utf-8"?>
<selector
  xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true" android:drawable="@drawable/start_b"/>
    <item android:state_pressed="false" android:drawable="@drawable/start_a"/>
</selector>
```

（5）为 main.xml 布局文件中的图片按钮设置 android:src 属性，其属性值是在步骤（4）中编写的 Drawable 资源。其关键代码如下：

```
android:src="@drawable/button_state"
```

（6）在 res\values 目录下的 strings.xml 文件中，添加字符串变量 artcle，用于保存游戏条款。其关键代码如下：

```
<string name="artcle">         温馨提示：本游戏适合各年龄段的玩家，请您合理安排游戏时间，不要沉迷游戏！
    当您连续在线 2 小时间后，系统将自动结束游戏。如果同意该条款请勾选“我同意”复选框，方可进入游戏。</string>
```



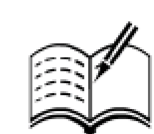
说明：

在 Android 中，空格使用“ ”表示。

（7）在主活动的 onCreate()方法中，获取布局文件中添加的“进入”图片按钮和“我同意”复选框，并为复选框添加状态改变监听器，用于实现当复选框被选中时显示“进入”按钮，否则不显示“进入”按钮。其具体代码如下：

```
final ImageButton imageButton=(ImageButton)findViewById(R.id.start);    //获取“进入”按钮
CheckBox checkbox=(CheckBox)findViewById(R.id.checkBox1);      //获取布局文件中添加的复选框
//为复选框添加监听器
checkbox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
     @Override
     public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
          if(isChecked){                                          //当复选框被选中
               imageButton.setVisibility(View.VISIBLE);           //设置“进入”按钮显示
          }else{
               imageButton.setVisibility(View.INVISIBLE);         //设置“进入”按钮不显示
          }
          imageButton.invalidate();                               //重绘 ImageButton
     }
});
```





（8）为“进入”按钮添加单击事件监听器，用于实现当用户单击“进入”按钮时，显示一个消息提示框。其具体代码如下：

```
imageButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                //显示消息提示框
                Toast.makeText(MainActivity.this, "进入游戏...", Toast.LENGTH_SHORT).show();
        }
});
```

6.6.2　应用相对布局管理器显示软件更新提示

【例 6.7】 在智能手机中，当系统中有软件更新时，经常会显示一个提示软件更新的界面。本实例中将应用相对布局管理器实现一个显示软件更新提示的界面。运行程序，显示如图 6.7 所示的运行效果。

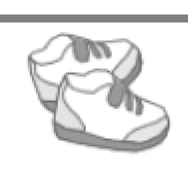
实例位置：光盘\MR\Instance\06\6.7



图 6.7　应用相对布局管理器显示软件更新提示

在 Eclipse 中创建 Android 项目，修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个 RelativeLayout 相对布局管理器，并且为其设置背景，最后在该布局管理器中，添加一个 TextView 和两个 Button，并设置它们的显示位置及对齐方式。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        android:id="@+id/relativeLayout1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:background="@drawable/background"
```



```
>
    <!-- 添加一个居中显示的文本视图 textView1 -->
    <TextView android:text="发现有 Widget 的新版本，您想现在就安装吗？"
    android:id="@+id/textView1"
    android:textSize="20px"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_centerInParent="true"
    />
    <!-- 添加一个在 button2 左侧显示的按钮 button1 -->
    <Button
    android:text="现在更新"
    android:id="@+id/button1"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_below="@+id/textView1"
    android:layout_toLeftOf="@+id/button2"
    />
    <!-- 添加一个按钮 button2，该按钮与 textView1 的右边界对齐 -->
    <Button
    android:text="以后再说"
    android:id="@+id/button2"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_alignRight="@+id/textView1"
    android:layout_below="@+id/textView1"
    />
</RelativeLayout>
```

说明：

上面的代码中，将文本视图 textView1 设置为在屏幕中央显示，然后设置按钮 button2 在 textView1 的下方居右边界对齐，最后设置按钮 button1 在 button2 的左侧显示。

6.7 本章常见错误

Android 程序编译时没有错误，但在运行时，却出现了下面的错误提示。

```
Unable to start activity ComponectInfo, java.lang.NullPointerException
```

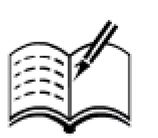
从上面的错误提示可以看出，该错误为空指针异常，分析原因，可能是以下两种情况造成的。

☑ 操作组件的代码放到了加载布局文件之前。

☑ 引用的布局文件不对。

针对以上两种情况，只需要将操作组件的代码放到加载布局文件之后，或者引用正确的布局文件即可。





6.8　本章小结

本章主要对 Android 界面设计时常用的 5 种布局管理器进行了讲解，主要包括线性布局、绝对布局、框架布局、相对布局和表格布局，在这 5 种布局管理器中，绝对布局只需要简单了解即可，而其他 4 种布局管理器在设计 Android 界面时经常用到，希望大家能够重点掌握。

6.9　跟我上机

参考答案：光盘\MR\跟我上机

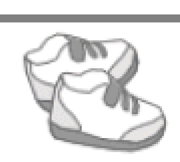
开发一个 Android 程序，要求使用线性布局管理器和相对布局管理器实现个性游戏开始界面，游戏的开始界面布局如图 6.8 所示。



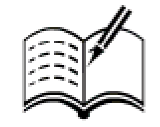
图 6.8　布局个性游戏开始界面布局

根据图 6.8 所示布局游戏开始界面需要用到线性布局管理器和相对布局管理器，并且需要借助 Android 中的 ImageView 组件。具体实现时，首先需要在 main.xml 布局文件中，将默认添加的布局代码中的 TextView 组件删除，然后添加一个 ImageView 组件，用于显示顶部图片，并设置其缩放方式为保持纵横比缩放图片让其完全覆盖 ImageView。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <!-- 添加顶部图片 -->
    <ImageView android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:layout_weight="1"
        android:src="@drawable/top" />
</LinearLayout>
```



然后，在 ImageView 组件的下方添加一个相对布局管理器，用于显示控制按钮，在该布局管理器中添加 5 个 ImageView 组件，并且第一个 ImageView 组件显示在相对布局管理器的中央，其他 4 个环绕在第一个组件的四周。其代码如下：



Note

```
<!-- 添加一个相对布局管理器 -->
<RelativeLayout android:layout_weight="2"
    android:layout_height="wrap_content"
    android:background="@drawable/bottom"
    android:id="@+id/relativeLayout1"
    android:layout_width="match_parent">
    <!-- 添中间位置的图片按钮 -->
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton0"
        android:src="@drawable/in"
        android:layout_alignTop="@+id/imageButton5"
        android:layout_centerInParent="true" />
    <!-- 添加上方显示的图片 -->
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton1"
        android:src="@drawable/setting"
        android:layout_above="@+id/imageButton0"
        android:layout_alignRight="@+id/imageButton0" />
    <!-- 添加下方显示的图片 -->
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton2"
        android:src="@drawable/exit"
        android:layout_below="@+id/imageButton0"
        android:layout_alignLeft="@+id/imageButton0" />
    <!-- 添加左侧方显示的图片-->
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton3"
        android:src="@drawable/help"
        android:layout_toLeftOf="@+id/imageButton0"
        android:layout_alignTop="@+id/imageButton0" />
    <!-- 添加右侧显示的图片 -->
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton4"
        android:src="@drawable/board"
        android:layout_toRightOf="@+id/imageButton0"
        android:layout_alignTop="@+id/imageButton0" />
</RelativeLayout>
```

第 7 章

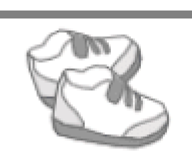
Android 程序调试与错误处理

（视频讲解：48 分钟）

开发 Android 程序时，不仅要注意程序代码的准确性与合理性，还要处理程序中可能出现的异常情况。Android SDK 中提供了 Log 类来获取程序的日志信息；另外，还提供了 LogCat 管理器用来查看程序运行的日志信息及错误日志。本章将详细讲解如何对 Android 程序进行调试及异常处理。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 使用 Log.d 方法输出 Debug 日志信息
- ☐ 使用 Log.e 方法输出错误日志信息
- ☐ 使用 Log.i 方法输出程序日志信息
- ☐ 使用 Log.v 方法输出冗余日志信息
- ☐ 使用 Log.w 方法输出警告日志信息
- ☐ 使用 try...catch 语句捕获 Android 程序异常
- ☐ 使用 throws 关键字抛出异常
- ☐ 使用 throw 关键字抛出异常
- ☐ 向 LogCat 视图中输出用户登录时间
- ☐ 使用 throw 关键字在方法中抛出异常





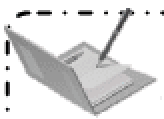
Note

7.1 输出日志信息的几种方法

Android SDK 中提供了 Log 类来获取程序运行时的日志信息，该类位于 android.util 命名空间中，它继承自 java.lang.Object 类。Log 类提供了一些方法，用来输出日志信息，其常用方法及说明如表 7.1 所示。

表 7.1 Log 类的常用方法及说明

方　法	说　明
d	输出 DEBUG（故障）日志信息
e	输出 ERROR（错误）日志信息
i	输出 INFO（程序）日志信息
v	输出 VERBOSE（冗余）日志信息
w	输出 WARN（警告）日志信息



说明：

表 7.1 中列出的 Log 类的相关方法都有多种重载形式，下面将介绍它们经常用到的重载形式。

7.1.1 Log.d 方法——输出故障日志信息

Log.d 方法用来输出 DEBUG（故障）日志信息，该方法有两种重载形式，其中开发人员经常用到的重载形式语法如下：

```
public static int v (String tag, String msg)
```

☑ tag：String 字符串，用来标识日志信息，它通常指定为可能出现 Debug 的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

【例 7.1】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中使用 Log.d 方法输出 Debug 日志信息的功能。

实例位置：光盘\MR\Instance\07\7.1

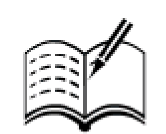
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 Button 组件。其主要代码如下：

```
<Button
    android:id="@+id/btn"
```



```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Debug 日志"
        />
```



Note

（2）打开 Activity 文件，首先根据 id 获取布局文件中的 Button 组件，然后为该组件设置单击监听事件，在监听事件中，使用 Log.d 方法输出 Debug 日志信息。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);              //获取 Button 组件
    btnButton.setOnClickListener(new OnClickListener() {          //设置监听事件
        @Override
        public void onClick(View v) {
            Log.d("DEBUG", "Debug 日志信息");                       //输出 DEBUG 日志信息
        }
    });
}
```

运行本实例，单击 Android 界面中的 Button 按钮，将会在 LogCat 管理器中看到如图 7.1 所示的结果。



L...	Time	PID	Application	Tag	Text
D	11-10 13:06:01.641	568	com.xiaoke.exam06...	DEBUG	Debug日志信息

图 7.1　使用 Log.d 方法输出 DEBUG 日志信息



说明：

使用 Log 类的相关方法输出的日志信息需要在 LogCat 管理器中查看，下面遇到时将不再提示。

7.1.2　Log.e 方法——输出错误日志信息

Log.e 方法用来输出 ERROR（错误）日志信息，该方法有两种重载形式，其中开发人员经常用到的重载形式语法如下：

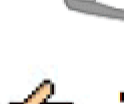
```
public static int e (String tag, String msg)
```

☑ tag：String 字符串，用来标识日志信息，它通常指定为可能出现错误的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

【例 7.2】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中使用 Log.e 方法输出错误日志信息的功能。





实例位置：光盘\MR\Instance\07\7.2

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 Button 组件。其主要代码如下：

```
<Button
    android:id="@+id/btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Error 日志"
    />
```

（2）打开 Activity 文件，首先根据 id 获取布局文件中的 Button 组件，然后为该组件设置单击监听事件，在监听事件中，使用 Log.e 方法输出错误日志信息。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);           //获取 Button 组件
    btnButton.setOnClickListener(new OnClickListener() {        //设置监听事件
        @Override
        public void onClick(View v) {
            Log.e("ERROR", "Error 日志信息");                   //输出 ERROR 日志信息
        }
    });
}
```

运行本实例，单击 Android 界面中的 Button 按钮，将会在 LogCat 管理器中看到如图 7.2 所示的结果。



图 7.2　使用 Log.e 方法输出错误日志信息

7.1.3　Log.i 方法——输出程序日志信息

Log.i 方法用来输出 INFO（程序）日志信息，该方法有两种重载形式，其中开发人员经常用到的重载形式语法如下：

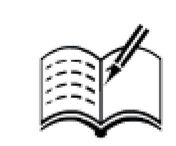
```
public static int i (String tag, String msg)
```

☑　tag：String 字符串，用来标识日志信息，它通常指定为类或者 Activity 的名称。

☑　msg：String 字符串，表示要输出的字符串信息。

【例 7.3】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中使用 Log.i 方法输出





Note

程序日志信息的功能。

实例位置：光盘\MR\Instance\07\7.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 Button 组件。其主要代码如下：

```
<Button
    android:id="@+id/btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="程序日志"
    />
```

（2）打开 Activity 文件，首先根据 id 获取布局文件中的 Button 组件，然后为该组件设置单击监听事件，在监听事件中，使用 Log.i 方法输出程序日志信息。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);            //获取 Button 组件
    btnButton.setOnClickListener(new OnClickListener() {         //设置监听事件
        @Override
        public void onClick(View v) {
            Log.i("INFO", "程序日志信息");                       //输出程序日志信息
        }
    });
}
```

运行本实例，单击 Android 界面中的 Button 按钮，将会在 LogCat 管理器中看到如图 7.3 所示的结果。

L...	Time	PID	Application	Tag	Text
I	11-10 13:39:19.812	713	com.xiaoke.exam06...	INFO	程序日志信息

标识为程序日志

图 7.3　使用 Log.i 方法输出程序日志信息

7.1.4　Log.v 方法——输出冗余日志信息

Log.v 方法用来输出 VERBOSE（冗余）日志信息，该方法有两种重载形式，其中开发人员经常用到的重载形式语法如下：

```
public static int v (String tag, String msg)
```

Note

☑ tag：String 字符串，用来标识日志信息，它通常指定为可能出现冗余的类或者 Activity 的名称。

☑ msg：String 字符串，表示要输出的字符串信息。

【例 7.4】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中使用 Log.v 方法输出冗余日志信息的功能。

实例位置：光盘\MR\Instance\07\7.4

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 Button 组件。其主要代码如下：

```
<Button
        android:id="@+id/btn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="冗余日志"
        />
```

（2）打开 Activity 文件，首先根据 id 获取布局文件中的 Button 组件，然后为该组件设置单击监听事件，在监听事件中，使用 Log.v 方法输出冗余日志信息。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);                //获取 Button 组件
    btnButton.setOnClickListener(new OnClickListener() {             //设置监听事件
        @Override
        public void onClick(View v) {
            Log.v("VERBOSE", "Verbose 日志信息");                     //输出冗余日志信息
        }
    });
}
```

运行本实例，单击 Android 界面中的 Button 按钮，将会在 LogCat 管理器中看到如图 7.4 所示的结果。

L...	Time	PID	Application	Tag	Text
V	11-10 13:45:41.422	758	com.xiaoke.exam06...	VERBOSE	Verbose日志信息

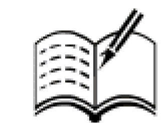
标识为 VERBOSE 日志

图 7.4　使用 Log.v 方法输出冗余日志信息

7.1.5　Log.w 方法——输出警告日志信息

Log.w 方法用来输出 WARN（警告）日志信息，该方法有 3 种重载形式，其中开发人员经常





用到的重载形式语法如下：

```
public static int w (String tag, String msg)
```

☑　tag：String 字符串，用来标识日志信息，它通常指定为可能出现警告的类或者 Activity 的名称。

☑　msg：String 字符串，表示要输出的字符串信息。

【例 7.5】 在 Eclipse 中创建 Android 项目，主要实现在 Android 程序中使用 Log.w 方法输出警告日志信息的功能。

实例位置：光盘\MR\Instance\07\7.5

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加一个 Button 组件。其主要代码如下：

```
<Button
    android:id="@+id/btn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Warn 日志"
    />
```

（2）打开 Activity 文件，首先根据 id 获取布局文件中的 Button 组件，然后为该组件设置单击监听事件，在监听事件中，使用 Log.w 方法输出警告日志信息。其代码如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);          //获取 Button 组件
    btnButton.setOnClickListener(new OnClickListener() {       //设置监听事件
        @Override
        public void onClick(View v) {
            Log.w("WARN", "Warn 日志信息");                      //输出 WARN 日志信息
        }
    });
}
```

运行本实例，单击 Android 界面中的 Button 按钮，将会在 LogCat 管理器中看到如图 7.5 所示的结果。



图 7.5　使用 Log.w 方法输出警告日志信息


Note

7.2　Android 程序调试

读者在程序开发过程中会不断体会到程序调试的重要性。为验证 Android 的运行状况，会经常在某个方法调用的开始和结束位置分别使用 Log.i 方法输出信息，并根据这些信息判断程序执行状况，这是非常古老的程序调试方法，而且经常导致程序代码混乱（导出的都是 Log.i 方法）。

本节将介绍使用 Eclipse 内置的 Java 调试器调试 Android 程序的方法，使用该调试器可以设置程序的断点、实现程序单步执行、在调试过程中查看变量和表达式的值等调试操作，这样可以避免在程序中编写大量的 Log.i 方法输出调试信息。

使用 Eclipse 的 Java 调试器需要设置程序断点，然后使用单步调试分别执行程序代码的每一行。

1．断点

设置断点是程序调试中必不可少的有效手段，Java 调试器每次遇到程序断点时都会将当前线程挂起，即暂停当前程序的运行。

可以在 Java 编辑器中显示代码行号的位置双击添加或删除当前行的断点，或者在当前行号的位置单击鼠标右键，在弹出的快捷菜单中选择“切换断点”命令实现断点的添加与删除，如图 7.6 所示。

2．程序调试

程序执行到断点被暂停后，可以通过“调试”视图工具栏上的按钮执行相应的调试操作，如运行、停止等。“调试”视图如图 7.7 所示。

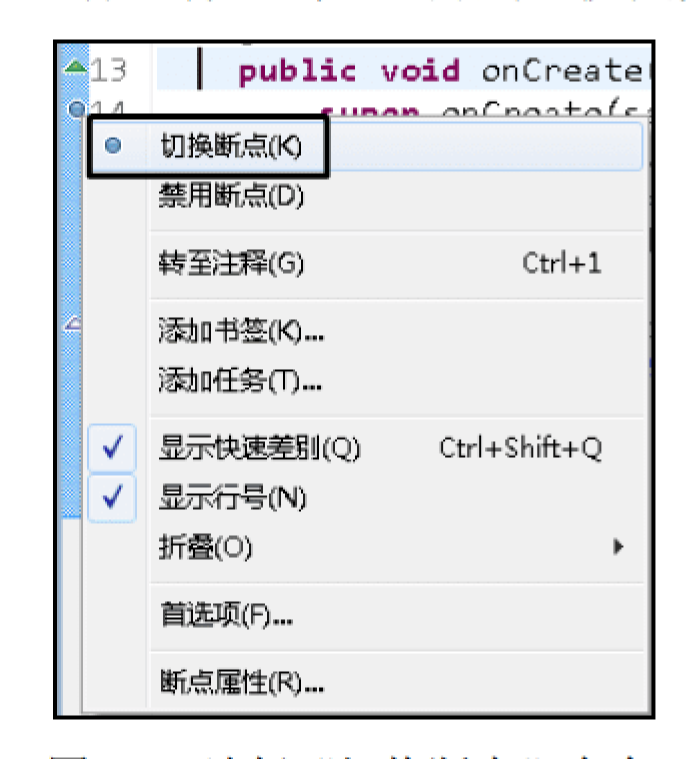
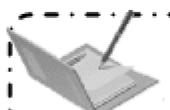


图 7.6　选择“切换断点”命令

图 7.7　“调试”视图

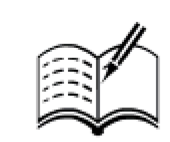
☑　单步跳过

在“调试”视图的工具栏中单击按钮或按 F6 键，将执行单步跳过操作，即运行单独的一行程序代码，但是不进入调用方法的内部，然后跳到下一个可执行点并暂挂线程。


说明：

不停地执行单步跳过操作，会每次执行一行程序代码，直到程序结束或等待用户操作。





☑　单步跳入

在“调试”视图的工具栏中单击按钮或按 F5 键，执行该操作将跳入调用方法或对象的内部单步执行程序并暂挂线程。

7.3　程序异常处理

为了保证程序有效地执行，需要对发生的异常进行相应的处理。在 Android 程序中，如果某个方法抛出异常，既可以在当前方法中进行捕捉，然后处理该异常，也可以将异常向上抛出，由方法调用者来处理。本节将向读者介绍 Android 中捕获异常的方法。

7.3.1　Android 程序出现异常怎么办

异常产生后，如果不做任何处理，程序就会被终止。例如，将一个字符串转换为整型，可以通过 Integer 类的 parseInt()方法来实现。但如果该字符串不是数字形式，parseInt()方法就会抛出异常，程序将停留在出现异常的位置，不再执行下面的语句。

【例 7.6】 在 Android 程序中将非字符型数值转换为 int 型，运行程序，系统会报出错误提示。

 **实例位置：光盘\MR\Instance\07\7.6**

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
int age = Integer.parseInt("20L");                                    //数据类型的转换
}
```

运行效果如图 7.8 所示。



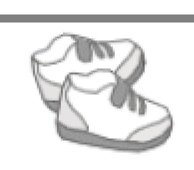
图 7.8　错误提示

在 LogCat 管理器中查看错误，可以看到如图 7.9 所示的结果。



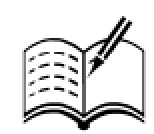
图 7.9　在 LogCat 管理器中查看错误

从图 7.9 中可以看出，本实例报出的是 NumberFormatException（字符串转换为数字）异常，



程序在执行类型转换代码时终止。

7.3.2　如何捕捉 Android 程序异常



Note

Android 程序中的异常捕获结构与 Java 类似，都是由 try、catch 和 finally 3 部分组成。其中，try 语句块存放的是可能发生异常的 Java 语句；catch 程序块在 try 语句块之后，用来激发被捕获的异常；finally 语句块是异常处理结构的最后执行部分，无论 try 块中的代码如何退出，都将执行 finally 块。

其语法如下：

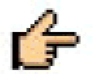
```
try{
    //程序代码块
}
catch(Exceptiontype1 e){
    //对 Exceptiontype1 的处理
}
catch(Exceptiontype2 e){
    //对 Exceptiontype2 的处理
}
...
finally{
    //程序块
}
```

通过异常处理器的语法可知，异常处理器大致分为 try…catch 语句块和 finally 语句块。

1．try…catch 语句

可将例 7.6 中的代码进行修改，使用 try…catch 语句捕获异常。

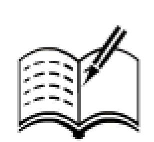
【例 7.7】 在例 7.6 的基础上，使用 try…catch 语句将可能出现的异常语句进行异常处理。

 **实例位置：光盘\MR\Instance\07\7.7**

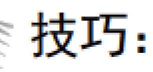
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    try {
        int age = Integer.parseInt("20L");                  //数据类型转换
     } catch (Exception e) {                                //catch 语句块用来获取异常信息
        e.printStackTrace();                                //输出异常
     }
}
```

上面的程序在运行时不会因为异常而终止，因为程序中将可能出现异常的代码用 try…catch 语句进行处理，当 try 代码块中的语句发生了异常时，程序就会调转到 catch 代码块中执行，执行完 catch 代码块中的程序代码后，继续执行 catch 代码块后的其他代码，而不会执行 try 代码块





中发生异常语句后面的代码。由此可知，Android 中的异常处理是结构化的，不会因为一个异常影响整个程序的执行。



注意：

Exception 是 try 代码块传递给 catch 代码块的变量类型，e 是变量名。catch 代码块中语句“e.getMessage();”用于输出错误性质。通常，异常处理常用以下 3 个函数来获取异常的有关信息。

☑ getMessage()函数：输出错误性质。
☑ toString()函数：给出异常的类型与性质。
☑ printStackTrace()函数：指出异常的类型、性质、栈层次及出现在程序中的位置。

技巧：

有时为了简单会忽略 catch 语句后的代码，这样 try…catch 语句就成了一种摆设，一旦程序在运行过程中出现了异常，就会导致最终运行结果与期望的不一致，而错误发生的原因很难查找。因此要养成良好的编程习惯，最好在 catch 代码块中有处理异常的代码。

2. finally 语句

完整的异常处理语句一定要包含 finally 语句，无论程序中有无异常发生，并且无论之前的 try…catch 是否顺利执行完毕，都会执行 finally 语句。

在以下 4 种特殊情况下，finally 块不会被执行。

☑ 在 finally 语句块中发生了异常。
☑ 在前面的代码中使用了 System.exit()退出程序。
☑ 程序所在的线程死亡。
☑ 关闭 CPU。

7.3.3　抛出异常的两种方法

若某个方法可能会发生异常，但不想在当前方法中处理这个异常，则可以使用 throws 和 throw 关键字在方法中抛出异常。下面分别介绍如何使用这两个关键字抛出异常。

1. 使用 throws 关键字抛出异常

throws 关键字通常被应用在声明方法时，用来指定方法可能抛出的异常。多个异常可使用逗号分隔。

【例 7.8】 在 Android 项目的 Activity 中创建 pop()方法，在该方法中抛出 NegativeArraySizeException 异常，然后在 onCreate()方法中调用 pop()方法，并实现异常处理。

实例位置：光盘\MR\Instance\07\7.8

```
//定义方法并抛出 NegativeArraySizeException 异常
static void pop() throws NegativeArraySizeException {
```

```
        int[] arr = new int[-3];                              //创建数组
}
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      try {                                                   //try 语句处理异常信息
            pop();                                            //调用 pop()方法
      } catch (NegativeArraySizeException e) {
            Log.i("EXCEPTION","pop()方法抛出的异常");          //输出异常信息
      }
}
```

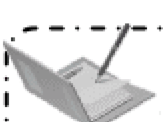
运行结果如图 7.10 所示。



L...	Time	PID	Application	Tag	Text
D	11-11 ...	614		dalvikvm	Debugger has detached;
I	11-11 ...	557	com.xiaoke.e...	EXCEPTION	pop()方法抛出的异常

图 7.10　使用 throws 关键字抛出异常

使用 throws 关键字将异常抛给上一级后，如果不想处理该异常，可以继续向上抛出，但最终要有能够处理该异常的代码。



说明：

如果是 Error、RuntimeException 或它们的子类，则可以不使用 throws 关键字来声明要抛出的异常，编译仍能顺利通过，但在运行时会被系统抛出。

2. 使用 throw 关键字抛出异常

throw 关键字通常用于方法体中，并且抛出一个异常对象。程序在执行到 throw 语句时立即终止，它后面的语句都不执行。通过 throw 关键字抛出异常后，如果想在上一级代码中来捕获并处理异常，则需要在抛出异常的方法中使用 throws 关键字在方法的声明中指明要抛出的异常；如果要捕捉 throw 抛出的异常，则必须使用 try…catch 语句。

throw 通常用来抛出用户自定义异常，下面通过实例介绍 throw 关键字的用法。

【例 7.9】 在 Eclipse 中创建 Android 项目，主要实现使用 throw 关键字抛出异常的功能。



实例位置：光盘\MR\Instance\07\7.9

程序的开发步骤如下：

（1）在项目中创建自定义异常类 MyException，继承类 Exception。其代码如下：

```
public class MyException extends Exception {                  //创建自定义异常类
            private static final long serialVersionUID = 1911857297231201652L;
            String message;                                   //定义 String 类型变量
      public MyException(String ErrorMessagr) {               //父类方法
            message = ErrorMessagr;
      }
```



```
    public String getMessage() {                     //覆盖 getMessage()方法
        return message;
    }
}
```

（2）在项目中创建 quotient()方法，该方法传递两个 int 型参数，如果其中的一个参数为负数，则抛出 MyException 异常。其代码如下：

```
int quotient(int x, int y) throws MyException {      //定义方法抛出异常
    if (y < 0) {                                     //判断参数是否小于 0
        throw new MyException("除数不能是负数");       //异常信息
    }
    return x / y;                                    //返回值
}
```

（3）在 onCreate 方法中捕捉异常，其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    try {                                            //try 语句包含可能发生异常的语句
        int result = quotient(3, -1);                //调用 quotient()方法
    } catch (MyException e) {                        //处理自定义异常
        Log.i("MYEXCEPTION",e.getMessage());         //输出异常信息
    } catch (ArithmeticException e) {                //处理 ArithmeticException 异常
        Log.i("ARITHMETICEXCEPTION","除数不能为 0");  //输出提示信息
    } catch (Exception e) {                          //处理其他异常
        Log.i("EXCEPTION","程序发生了其他的异常");     //输出提示信息
    }
}
```

运行结果如图 7.11 所示。



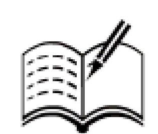
L...	Time	PID	Application	Tag	Text
D	11-11 ...	35		dalvikvm	GC_EXPLICIT fr
I	11-11 ...	658	com.xiaoke.e...	MYEXCEPTION	除数不能是负数

图 7.11　使用 throw 关键字抛出自定义异常



说明：

在上面的实例中使用了多个 catch 语句来捕捉异常。如果调用"quotient(3,-1)"，将发生 MyException 异常，程序调转到"catch (MyException e)"代码块中执行；如果调用"quotient(5,0)"，会发生 ArithmeticException 异常，程序跳转到"catch (ArithmeticException e)"代码块中执行；若还有其他异常发生，将使用"catch (Exception e)"捕捉异常。由于 Exception 是所有异常类的父类，如果将"catch (Exception e)"代码块放在其他两个代码块的前面，后面的代码块将永远得不到执行，也就没有什么意义了，所以 catch 语句的顺序不可调换。


Note

7.3.4 何时使用异常处理

Android 异常强制用户去考虑程序的强健性和安全性。异常处理不应用来控制程序的正常流程，其主要作用是捕获程序在运行时发生的异常并进行相应的处理。编写代码时处理某个方法可能出现的异常，可遵循以下几条原则。

☑ 在当前方法声明中使用 try...catch 语句捕获异常。

☑ 一个方法被覆盖时，覆盖它的方法必须抛出相同的异常或异常的子类。

☑ 如果父类抛出多个异常，则覆盖方法必须抛出那些异常的一个子集，不能抛出新异常。

7.4 综 合 应 用

7.4.1 向 LogCat 视图中输出用户登录时间

【例 7.10】 本实例要求在屏幕中添加一个“用户登录”按钮，单击该按钮，向 LogCat 视图中输出用户的登录时间，效果如图 7.12 所示。

实例位置：光盘\MR\Instance\07\7.10

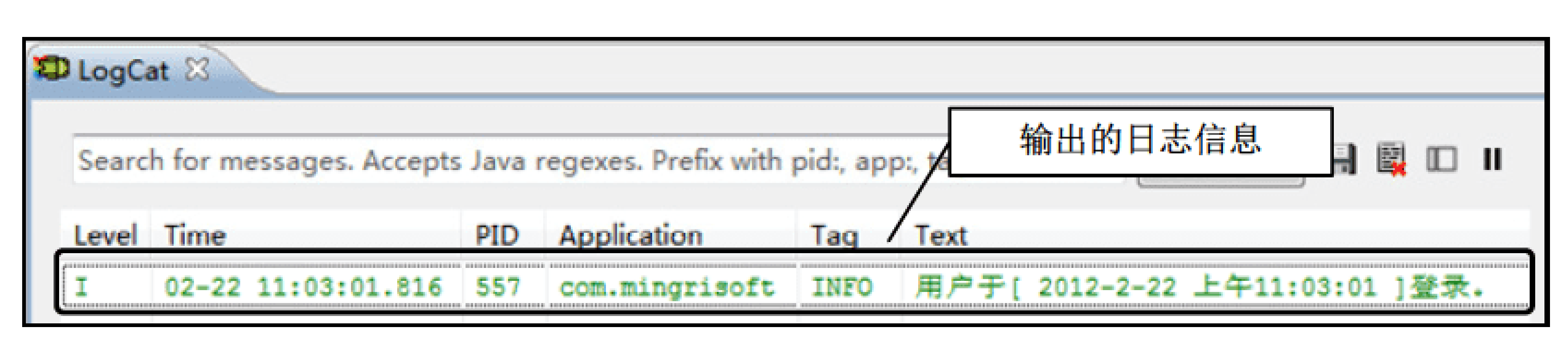


图 7.12 向 LogCat 视图中输出用户登录时间

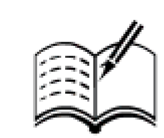
本实例实现时主要用到了 Log.i 方法，其代码如下：

```
Button btnButton = (Button) findViewById(R.id.button1);          //获取 Button 组件
btnButton.setOnClickListener(new OnClickListener() {             //设置监听事件
	@Override
	public void onClick(View v) {
	//输出程序日志信息
		Log.i("INFO", "用户于[ "+new Date().toLocaleString()+" ]登录。");
	}
});
```

7.4.2 使用 throw 关键字在方法中抛出异常

【例 7.11】 在项目开发中，通常是自上向下进行的，在完成项目的整体设计后，需要对每





个接口和类进行编写。如果一个类使用了其他类还没有实现的方法，则可以在实现其他类方法时让其抛出 UnsupportedOperationException 异常，以便在以后进行修改完成。本实例要求使用 throw 关键字在方法中抛出“方法尚未实现”异常，效果如图 7.13 所示。

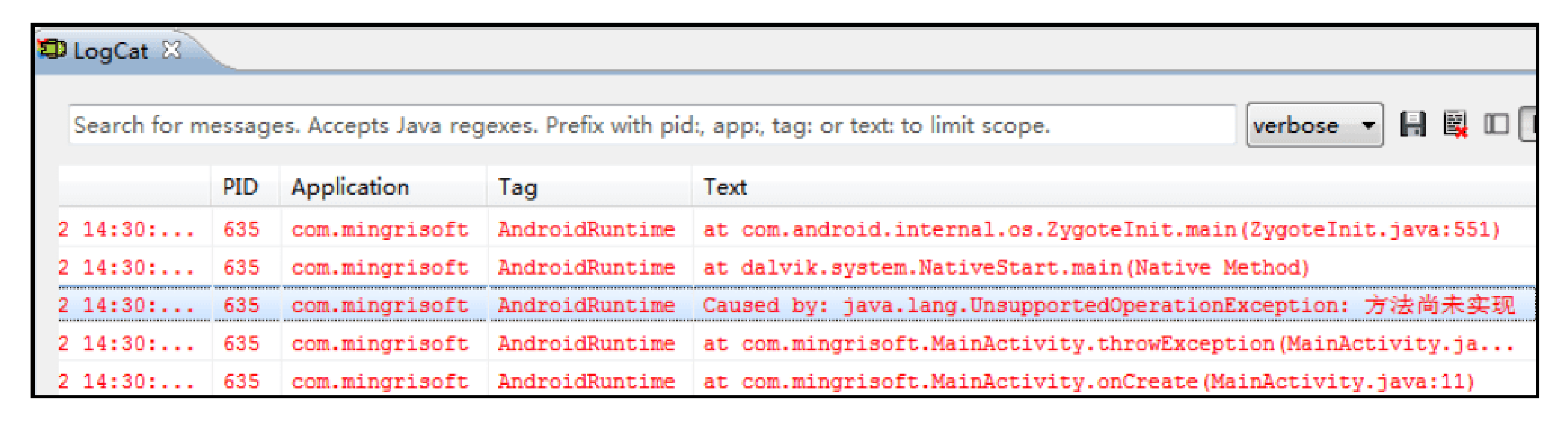


图 7.13　使用 throw 关键字在方法中抛出异常

实例位置：光盘\MR\Instance\07\7.11

本实例主要使用 throw 关键字实现在方法中抛出 UnsupportedOperationException 异常，其代码如下：

```
public class MainActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        throwException();                                          //调用抛出异常的方法
    }
    public static void throwException() {
        throw new UnsupportedOperationException("方法尚未实现");    //抛出异常
    }
}
```

7.5　本章常见错误

try 语句块主要用来捕获程序运行时的异常，catch 语句块用来处理异常，或者说将发生异常时要执行的代码括起来，但无论是否有异常，最后一定会执行 finally 语句块中的代码。那么，如果在 try 语句块中使用了 return 语句，finally 中的语句会不会执行呢？

例如，在 try 语句块中使用 return 语句，其代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    try{
        int i = 5;
```

```
            Object obj = i;                          //整型变量装箱操作
            Log.i("INFO","类型转换成功！");
            return;                                  //使用 return 语句
        }
        catch (Exception ex){                        //这里是处理异常的语句块
            Log.i("INFO", ex.getMessage());          //输出异常信息
        }
        finally{                                     //finally 语句
            Log.i("INFO","必须执行 finally 语句块");    //若执行 finally 语句，则输出此信息
        }
}
```

上面代码的运行效果如图 7.14 所示。

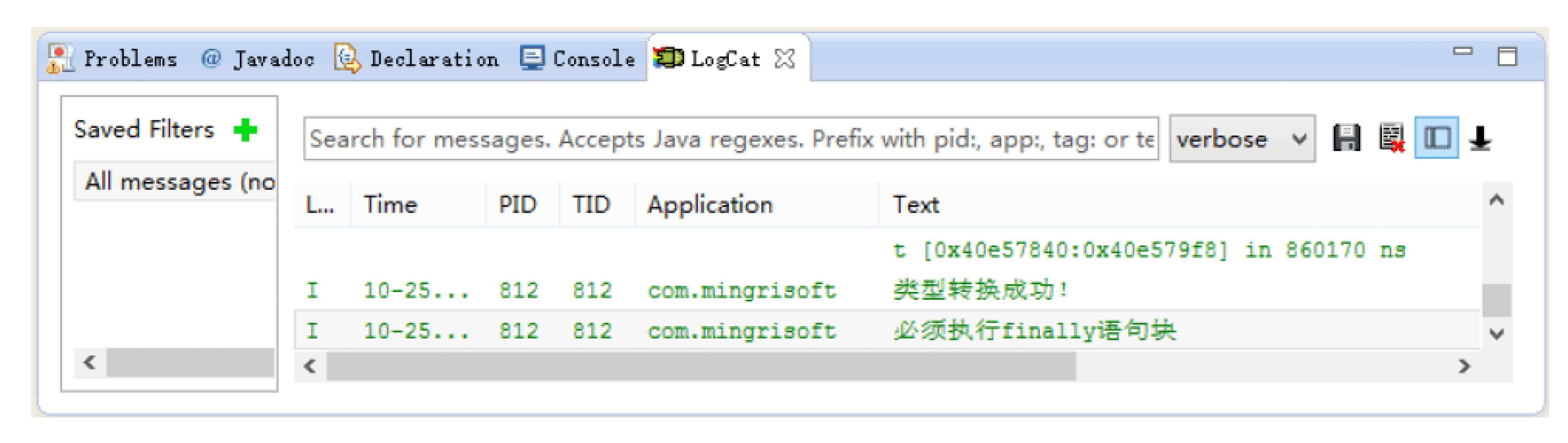


图 7.14　在 try 语句块中使用 return 语句

从上面的运行结果可以看到，即使在 try 语句块中使用 return 语句，也仍然阻止不了 finally 中语句的执行。

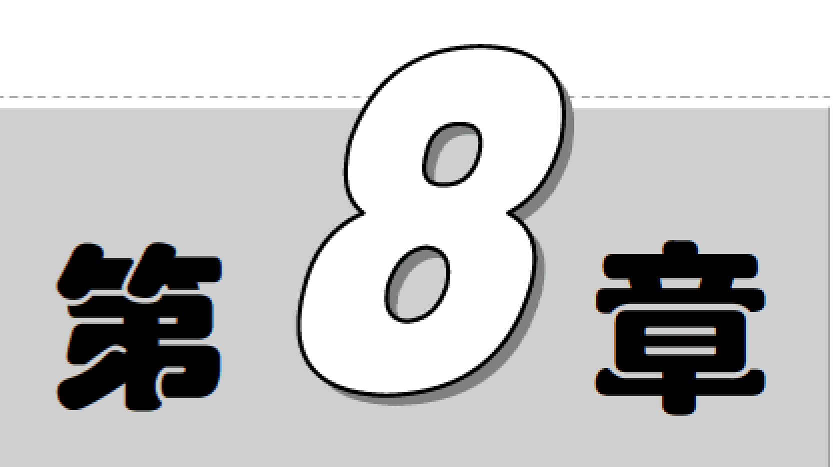
7.6　本 章 小 结

本章向读者介绍的是 Android 中的程序调试及异常处理机制。通过本章的学习，读者应该熟练掌握使用 Log 类输出日志信息的几种方法，并能够通过 LogCat 管理器查看日志信息；另外，读者应该熟悉常见的几种程序调试操作，并熟练掌握 Android 的异常处理机制。Android 中的异常处理与 Java 类似，都是通过 try…catch 语句来实现的，也可以使用 throws 语句向上抛出。建议读者不要将异常抛出，而应该编写合理的异常处理语句。对于异常处理的使用原则，读者也应该熟悉。

7.7　跟 我 上 机

参考答案：光盘\MR\跟我上机

开发一个 Android 程序，要求使用 Log.w 方法向 LogCat 视图中输出警告日志信息“系统内存不足”。该程序实现时，需要在布局文件中添加一个 Button 组件，id 属性设置为“@+id/btn”，text 属性设置为“Warn 日志”。



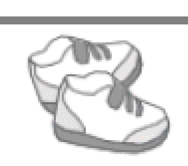
Activity 的使用

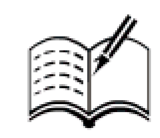
（视频讲解：1 小时 44 分钟）

Activity 是 Android 系统中最基本也是最为常用的组件。在一个 Android 程序中，一个 Activiy 通常就是一个单独的屏幕。本章将对 Activity 的使用及其生命周期进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- □ 在 AndroidManifest.xml 主设置文件中配置 Activity
- □ 在程序中演示 Activity 的生命周期
- □ 新建一个 Activity
- □ 启动一个或多个 Activity
- □ 在多个 Activity 之间实现相互传值
- □ 关闭 Activity
- □ 根据输入的生日判断星座
- □ 带选择头像的用户注册界面
- □ 仿 QQ 客户端登录界面
- □ 实现一个泡泡龙游戏的关于功能




Note

8.1 Activity 入门

Activity 是 Android 系统提供的一个可视的用户交互接口，所有和用户的交互都发生在这里（类似于 Windows 的窗口）。Activity 在创建时生成各种控件视图（View），这些视图负责具体功能，如 TextView、Button 等。Activity 通常使用全屏模式，也有浮动窗口模式（通过设置属性 windowIsFloating）和嵌入模式。本节将对 Activity 进行详细讲解。

8.1.1 Activity 概述

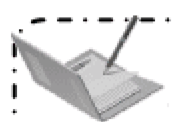
Activity 是 Android 程序中最基本的模块，它是为用户操作而展示的可视化用户界面，一个 Android 应用程序中可以只有一个 Activity，也可以包含多个，每个 Activity 的作用及其数目取决于应用程序及其设计。例如，可以使用一个 Activity 展示一个菜单项列表供用户选择，也可以显示一些包含说明的照片等。

在 Android 程序中，每个 Activity 都被给予一个默认的窗口以进行绘制，一般情况下，这个窗口是满屏的，但它也可以是一个小的、位于其他窗口之上的浮动窗口。

技巧：

一个 Activity 也可以使用超过一个的窗口——例如，在 Activity 运行过程中弹出的一个供用户反应的小对话框，或者，当用户选择了屏幕上特定项目后显示的必要信息。

Activity 窗口显示的可视内容是由一系列视图构成的，这些视图均继承自 View 基类。每个视图均控制着窗口中一块特定的矩形空间，父级视图包含并组织其子视图的布局，而底层视图则在它们控制的矩形中进行绘制，并对用户操作做出响应，所以，视图是 Activity 与用户进行交互的界面。例如，开发人员可以通过视图显示一个图片，然后在用户单击它时产生相应的动作。



说明：

Android 中有很多既定的视图供开发人员直接使用，如按钮、文本域、卷轴、菜单项和复选框等。

8.1.2 Activity 的 4 种状态

Activity 作为 Android 应用程序最重要的一部分，它主要有 4 种状态，分别如下。

- ☑ Runing 状态：一个新 Activity 启动入栈后，它在屏幕最前端，处于栈的最顶端，此时它处于可见并可与用户交互的激活状态。如图 8.1 所示为一个 Activity 的 Runing 状态。
- ☑ Paused 状态：当 Activity 被另一个透明或者 Dialog 样式的 Activity 覆盖时的状态，此时它依然与窗口管理器保持连接，系统继续维护其内部状态，所以它仍然可见，但它已经





Note

失去了焦点，故不可与用户交互。如图 8.2 所示为一个 Activity 的 Paused 状态。

- ☑ Stopped 状态：当 Activity 不可见时，Activity 处于 Stopped 状态。Activity 将继续保留在内存中保持当前的所有状态和成员信息，假设系统别的地方需要内存的话，这时它是被回收对象的主要候选。当 Activity 处于 Stopped 状态时，一定要保存当前数据和当前的 UI 状态，否则一旦 Activity 退出或关闭时，当前的数据和 UI 状态就丢失了。
- ☑ Killed 状态：Activity 被杀掉以后或者被启动以前，处于 Killed 状态。这时 Activity 已被移除 Activity 堆栈中，需要重新启动才可以显示和使用。



图 8.1 Activity 的 Runing 状态



图 8.2 Activity 的 Paused 状态



说明：

Android 的 4 种状态中，Runing 和 Paused 状态是可见的，而 Stopped 和 Killed 状态是不可见的。

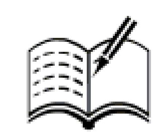
8.1.3 Activity 的属性

在 Android 中，Activity 是作为一个对象存在的，因此，它与 Android 中的其他对象类似，也支持很多 XML 属性。Activity 支持的常用 XML 属性如表 8.1 所示。

表 8.1 Activity 支持的 XML 属性

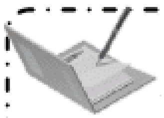
XML 属性	描述
android:name	指定 Activity 对应的类名
android:theme	指定应用什么主题
android:label	设置显示的名称，一般在 Launcher 里面显示
android:icon	指定显示的图标，在 Launcher 里面显示
android:screenOrientation	指定当前 Activity 显示横竖等
android:allowTaskReparenting	是否允许 Activity 更换从属的任务，如从短信息任务切换到浏览器任务





续表

XML 属性	描　述
android:alwaysRetainTaskState	当用户离开一个 Task 一段时间后，系统就会清理掉 Task 里除了根 Activity 以外的 Activity，如果一个 Task 里的根 Activity 的 alwaysRetainTaskState 属性设置为 true，那么前面描述的默认情况就不会出现了，Task 即使过了一段时间也会一直保留所有的 Activity
android:clearTaskOnLaunch	根 Activity 为 true，用户离开 Task 并返回时，Task 会清除直到根 Activity
android:configChanges	当配置 list 发生修改时，是否调用 onConfigurationChanged()方法
android:excludeFromRecents	是否可被显示在最近打开的 Activity 列表里
android:exported	是否允许 activity 被其他程序调用
android:launchMode	设置 Activity 的启动方式 standard、singleTop、singleTask 和 singleInstance，其中前两个为一组，　后两个为一组
android:finishOnTaskLaunch	是否关闭已打开的 Activity，当用户重新启动这个任务时
android:noHistory	当用户切换到其他屏幕时，是否需要移除这个 Activity
android:taskAffinity	Activity 的亲属关系，默认情况同一个应用程序下的 Activity 有相同的关系
android:process	一个 Activity 运行时所在的进程名，所有程序组件运行在应用程序默认的进程中，这个进程名跟应用程序的包名一致
android:windowSoftInputMode	定义软键盘弹出的模式


说明：

android:noHistory 属性是从 API level 3 开始引入的。

【例 8.1】 在 Eclipse 中创建一个 Android 程序，程序会自动在 AndroidManifest.xml 主设置文件中配置默认的 Activity。其代码如下：

实例位置：光盘\MR\Instance\08\8.1

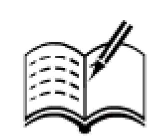
```
<activity
    android:label="@string/app_name"
    android:name=".MainActivity" >
    <intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

上面的代码中，用到了 Activity 的 android:label 和 android:name 属性，其中，android:label 属性指定了 Activity 的显示标题，而 android:name 属性指定了 Activity 对应的类名；另外，在配置默认 Activity 时，用到了<intent-filter>属性，该属性用来设置 Activity 作为程序入口，并且显示在启动栏中，其中的<action android:name="android.intent.action.MAIN"/>属性用来定义 Activity 作为 Android 应用程序的入口，而<category android:name="android.intent.category.LAUNCHER"/>属性用来指定 Activity 显示在 Launcher 里。



说明：

在 AndroidManifest.xml 文件中设置 Activity 时，只有设置了<category android:name= "android.intent.category.LAUNCHER"/>属性，Activity 才能显示在屏幕的启动栏中。



技巧：

如果一个 Android 程序中有多个 Activity，可以在 AndroidManifest.xml 文件中通过<intent-filter>属性设置默认启动的 Activity，该属性类似于 Java 代码中的 Main 函数。

8.2　Activity 的生命周期

开发程序时，生命周期是大部分对象都需要考虑的一个问题，对象的生命周期一般都是从创建开始，到销毁结束，而 Activity 作为 Android 程序中的一个核心窗口对象，在使用时，有其特殊的生命周期。本节将对 Activity 的生命周期进行详细讲解。

8.2.1　Activity 生命周期概述

Android 程序创建时，系统会自动在其.java 源文件中重写 Activity 类的 onCreate()方法，该方法是创建 Activity 时必须调用的一个方法，另外，Activity 类中还提供了如 onStart()、onResume()、onPause()、onStop()和 onDestroy()等方法，这些方法的先后执行顺序构成了 Activity 对象的一个完整生命周期。图 8.3 是 Android 官方给出的 Activity 对象生命周期图。

图 8.3 所示的 Activity 对象生命周期图中涉及了 onCreate()、onStart()、onResume()、onPause()、onStop()和 onDestroy()等 7 个方法，这 7 个方法定义了 Activity 的完整生命周期，而该完整生命周期又可以分成 3 个嵌套生命周期循环，分别如下。

- ☑ 前台生命周期：自 onResume()方法调用起，至相应的 onPause()方法调用为止。在此期间，Activity 位于前台最上面并与用户进行交互，Activity 会经常在暂停和恢复之间进行状态转换，例如，当设备转入休眠状态或者有新的 Activity 启动时，将调用 onPause()方法，而当 Activity 获得结果或者接收到新的 Intent 时，会调用 onResume()方法。
- ☑ 可视生命周期：自 onStart()方法调用开始，直到相应的 onStop()方法调用结束。在此期间，用户可以在屏幕上看到 Activity，尽管它也许并不是位于前台或者也不与用户进行交互。在这两个方法之间，可以保留用来向用户显示这个 Activity 所需的资源。例如，当用户看不到显示的内容时，可以在 onStart()方法中注册一个 BroadcastReceiver 广播接收器来监控可能影响 UI 的变化，而在 onStop()方法中来注消。onStart()和 onStop()方法可以随着应用程序是否被用户可见而被多次调用。
- ☑ 完整生命周期：自第一次调用 onCreate()方法开始，直至调用 onDestroy()方法为止。Activity 在 onCreate()方法中设置所有“全局”状态以完成初始化，而在 onDestroy()方法中释放所有系统资源。例如，如果 Activity 有一个线程在后台运行从网络上下载数据，



Note

它会在 onCreate()方法中创建线程，而在 onDestroy()方法中销毁线程。

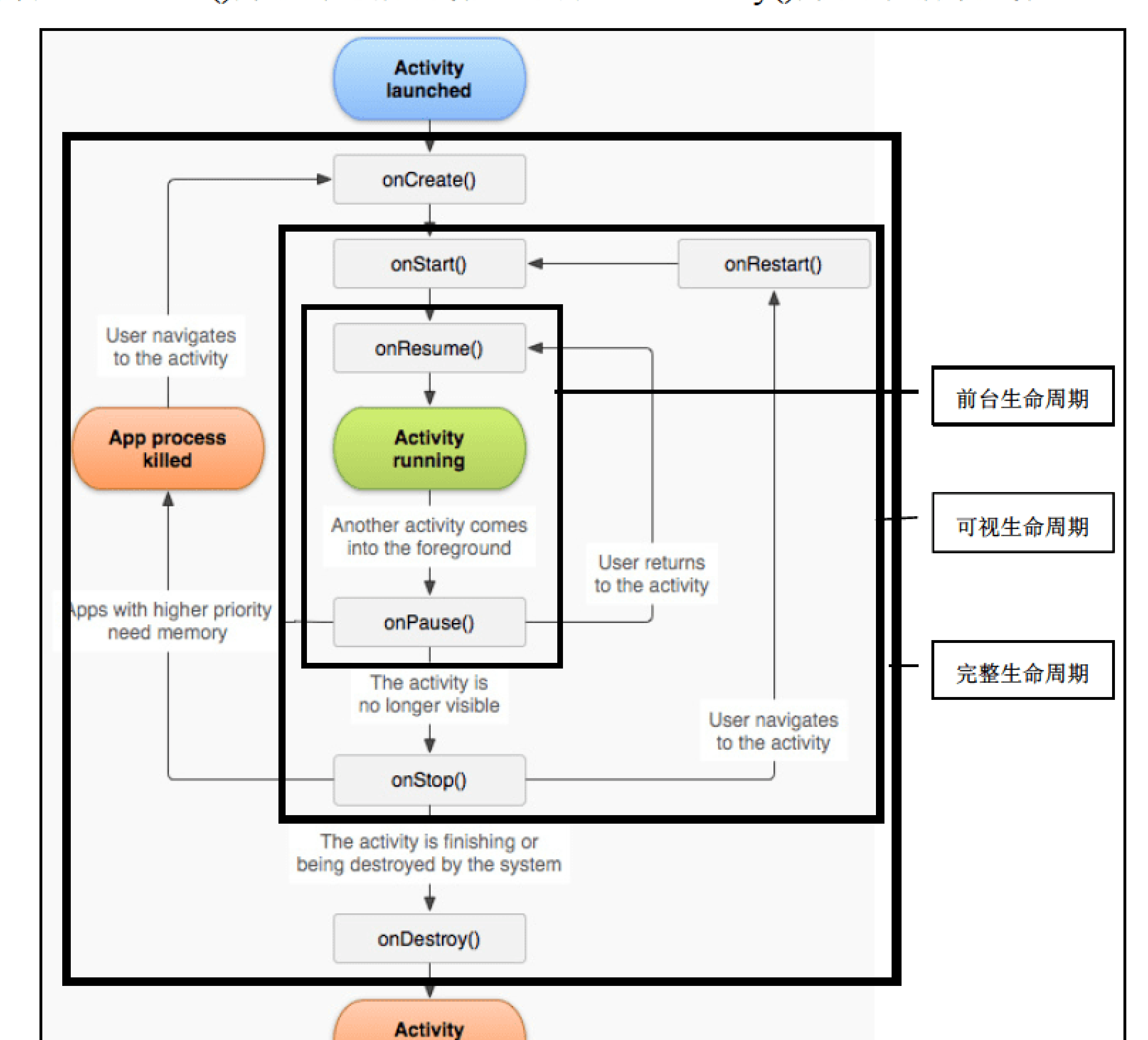


图 8.3　Activity 对象生命周期

8.2.2　Activity 的方法

8.2.1 节讲解了 Activity 的生命周期，其中主要涉及了 onCreate()、onStart()、onResume()、onPause()、onStop()、OnRestart()和 onDestroy() 7 个方法，下面分别对这 7 个方法进行介绍。

1．onCreate()方法

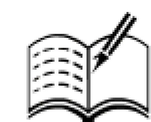
onCreate()方法用来创建 Activity，其覆写形式如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
}
```

技巧：

覆写 Activity 的相关方法时，可以通过 Eclipse 中的“源代码” / “覆盖/实现方法”命令实现。其具体步骤为：选择该命令，在弹出的“覆盖/实现方法”对话框（如图 8.4 所示）中选中要覆写方法前面的复选框，单击“确定”按钮即可。





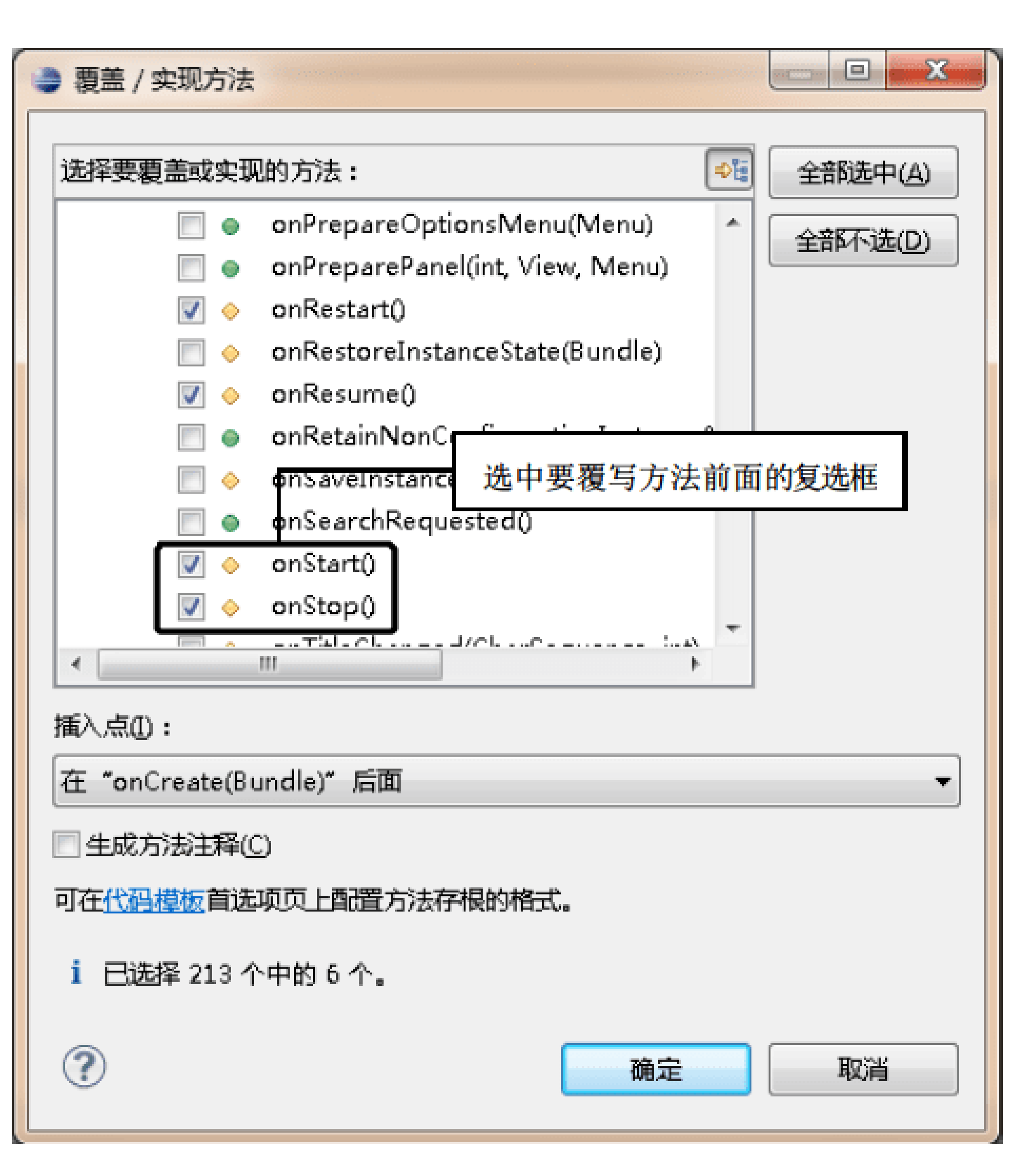



图 8.4 “覆盖/实现方法”对话框

2．onStart()方法

onStart()方法用来启动 Activity，其覆写形式如下：

```
@Override
protected void onStart() {
	//TODO Auto-generated method stub
	super.onStart();
}
```

3．onResume()方法

onResume()方法用来恢复 Activity，其覆写形式如下：

```
@Override
protected void onResume() {
	//TODO Auto-generated method stub
	super.onResume();
}
```

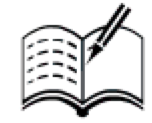
4．onPause()方法

onPause()方法用来暂停 Activity，其覆写形式如下：

```
@Override
protected void onPause() {
```



```
        //TODO Auto-generated method stub
        super.onPause();
}
```



5．onStop()方法

onStop()方法用来停止 Activity，其覆写形式如下：

```
@Override
protected void onStop() {
        //TODO Auto-generated method stub
        super.onStop();
}
```

6．onRestart()方法

onRestart()方法用来重启 Activity，其覆写形式如下：

```
@Override
protected void onRestart() {
        //TODO Auto-generated method stub
        super.onRestart();
}
```

7．onDestroy()方法

onDestroy()方法用来销毁 Activity，其覆写形式如下：

```
@Override
protected void onDestroy() {
        //TODO Auto-generated method stub
        super.onDestroy();
}
```

【例 8.2】 在 Eclipse 中创建 Android 项目，主要通过 onCreate()、onStart()、onResume()、onPause()、onStop()、OnRestart()和 onDestroy() 7 个方法的调用，演示 Activity 的生命周期。

实例位置：光盘\MR\Instance\08\8.2

在创建 Android 项目时，由于 onCreate()方法默认进行覆写，所以在创建的 Android 项目中覆写另外 6 个方法，即 onStart()、onResume()、onPause()、onStop()、OnRestart()和 onDestroy()，并分别在这 7 个方法的方法体内使用 Log.i 方法输出对应的方法名。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {                    //创建 Activity
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Log.i("ACTIVITY", " MainActivity ==)onCreate");
}
@Override
```





```
protected void onDestroy() {                                    //销毁 Activity
        //TODO Auto-generated method stub
        super.onDestroy();
        Log.i("ACTIVITY", " MainActivity ==》 onDestroy");
}
@Override
protected void onPause() {                                      //暂停 Activity
        //TODO Auto-generated method stub
        super.onPause();
        Log.i("ACTIVITY", " MainActivity==》 onPause");
}
@Override
protected void onRestart() {                                    //重启 Activity
        //TODO Auto-generated method stub
        super.onRestart();
        Log.i("ACTIVITY", " MainActivity ==》 onRestart");
}
@Override
protected void onResume() {                                     //恢复 Activity
        //TODO Auto-generated method stub
        super.onResume();
        Log.i("ACTIVITY", " MainActivity ==》 onResume");
}
@Override
protected void onStart() {                                      //启动 Activity
        //TODO Auto-generated method stub
        super.onStart();
        Log.i("ACTIVITY", " MainActivity ==》 onStart");
}
@Override
protected void onStop() {                                       //停止 Activity
        //TODO Auto-generated method stub
        super.onStop();
        Log.i("ACTIVITY", " MainActivity ==)onStop");
}
```

运行本实例，当第一次运行后，在 LogCat 管理器中看到如图 8.5 所示的日志信息。

Level	Time	PID	TID	Application	Tag	Text
I	07-19 17:01:34.135	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onCreate
I	07-19 17:01:34.135	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onStart
I	07-19 17:01:34.135	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onResume

图 8.5　第一次运行 Android 程序时的日志信息

当用户单击 Android 模拟器上的返回按钮时，退出该 Android 程序，并在 LogCat 管理器中显示如图 8.6 所示的日志信息。

Level	Time	PID	TID	Application	Tag	Text
I	07-19 17:05:04.905	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onPause
I	07-19 17:05:07.755	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onStop
I	07-19 17:05:07.755	831	831	com.xiaoke.exam08...	ACTIVITY	MainActivity==》 onDestroy

图 8.6　退出 Android 程序时的日志信息





Note

说明：

在图 8.5 和图 8.6 所示的日志信息中，没有看到 onRestart()方法的运行，这是因为该程序中只有一个 Activity，无法在程序中进行返回操作，所以不能执行 onRestart()方法；如果一个 Android 程序中有多个 Activity，当在这些 Activity 之间进行切换时，即可看到 onRestart()方法的执行过程。

8.3 Activity 常用操作

Activity 是 Android 程序中最基本的一个对象，该对象有一些常见的操作，如创建、启动、关闭和多个 Activity 间的传值等。本节将对 Activity 的常见操作进行详细讲解。

8.3.1 创建 Activity

在创建 Android 程序时，系统会自动创建一个默认的 Activity，但是，如何手动创建 Activity 呢？下面进行详细介绍。

手动创建 Activity 的步骤如下：

（1）在加载了 Android 程序的 Eclipse 中，选择“文件”/“新建”/“类”命令，或者单击工具栏中的按钮，弹出如图 8.7 所示的“新建 Java 类”对话框。

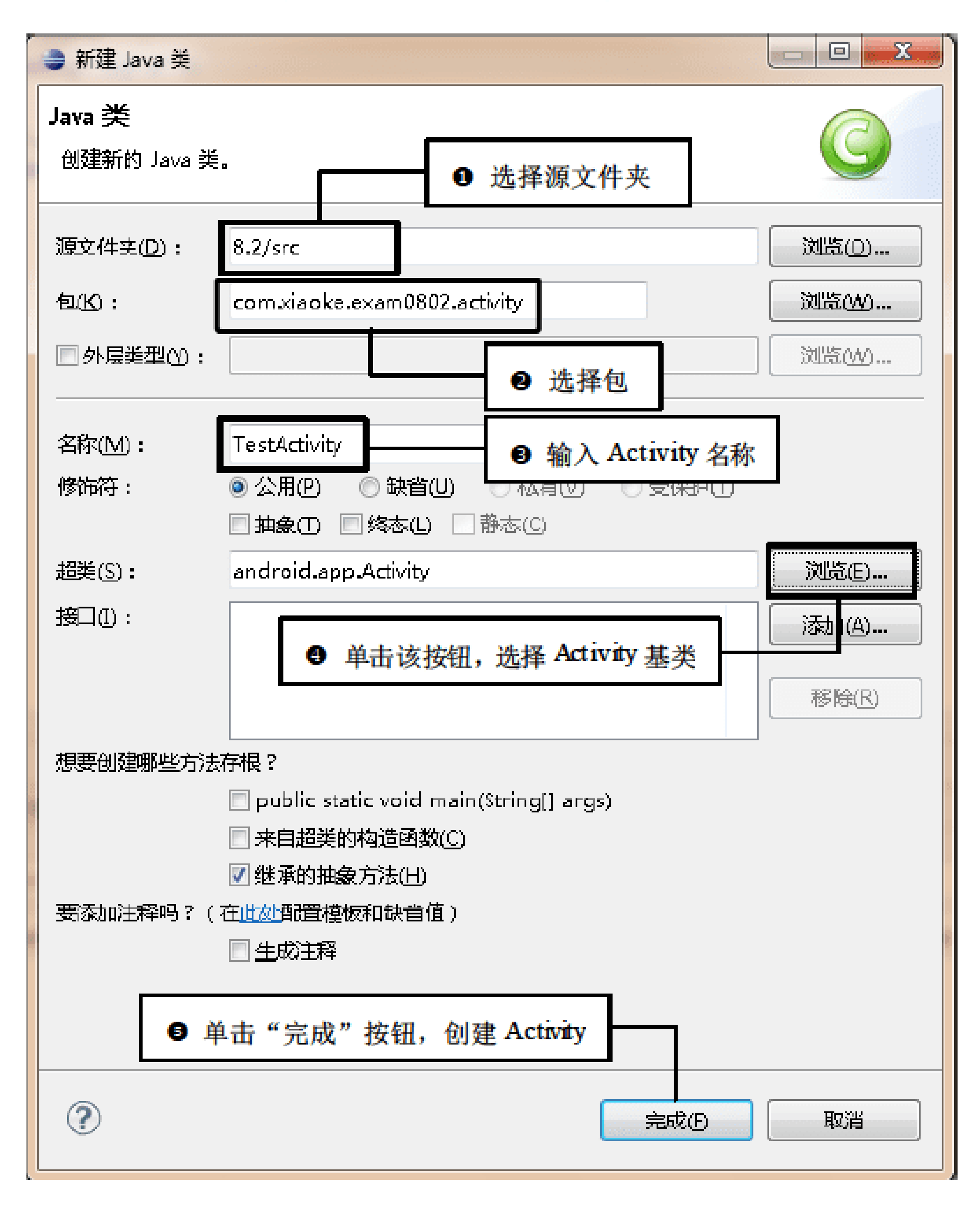
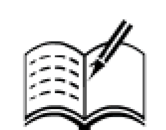


图 8.7 “新建 Java 类”对话框





（2）在该对话框中首先选择源文件夹、包，并输入 Activity 名称，然后单击“超类”文本框后面的“浏览”按钮，在弹出的“选择超类”对话框中找到 android.app.Activity 基类，如图 8.8 所示。

（3）单击“选择超类”对话框中的“确定”按钮，返回“新建 Java 类”对话框，单击“完成”按钮，即可创建一个 Activity，创建完成的 Activity 及其代码如图 8.9 所示。

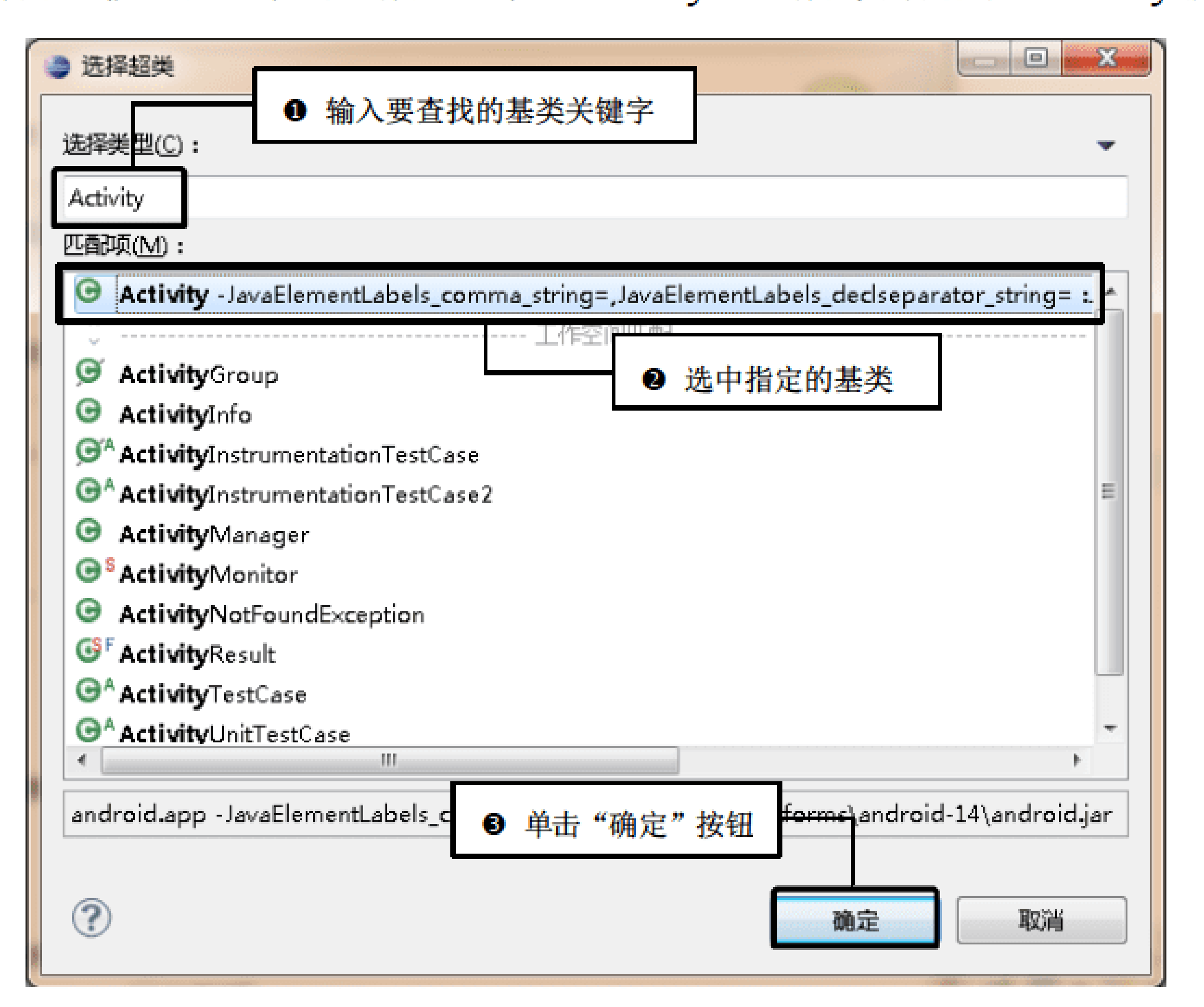




图 8.8　“选择超类”对话框

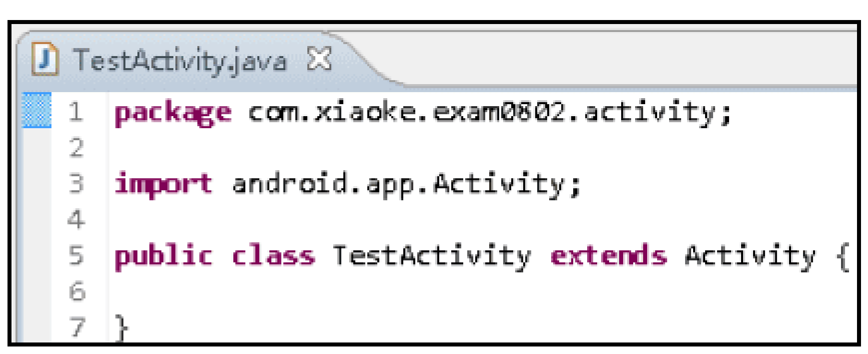


图 8.9　创建完成的 Activity 及其代码

8.3.2　启动一个或多个 Activity

创建完 Activity 后，需要通过执行相关事件启动 Activity，而且启动 Activity 之前，首先需要在 AndroidManifest.xml 主设置文件中对要启动的 Activity 进行配置。例如，这里配置 TestActivity 的代码如下：

```
<activity
    android:label="@string/app_name"
    android:name=".TestActivity" >
</activity>
```

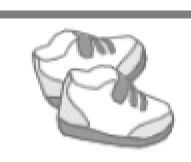
说明：

这里在配置 TestActivity 时，没有使用<intent-filter>属性，因此，该 Activity 不作为 Android 程序的默认启动项。

配置完 Activity 之后，可以通过关联的事件启动 Activity，在关联事件中可以通过 startActivity() 或者 startActivities()方法实现，下面分别对这两个方法进行介绍。

1．startActivity()方法

startActivity()方法用来启动单个 Activity，其语法格式如下：



```
public void startActivity(Intent intent)
```

参数 intent 表示要启动的 Intent 对象。



Note

说明：

Intent（意图）是一个对象，它是一个被动的数据结构保存一个将要执行的操作的抽象描述，或在广播的情况下，通常是某事已经发生并正在执行，开发人员通常使用该对象激活 Activity、Service 和 BroadcastReceiver。有关 Intent 对象的内容，将在第 9 章进行详细讲解。

【例 8.3】 在 Eclipse 中创建 Android 项目，主要实现使用 startActivity()方法启动一个 Activity 的功能。

实例位置：光盘\MR\Instance\08\8.3

程序的开发步骤如下：

（1）按照 8.3.1 节中讲解的步骤创建一个 Activity，命名为 TestActivity，并在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btnStart，并设置其文本为“启动 Activity”。其代码如下：

```
<Button
    android:id="@+id/btnStart"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="启动 Activity"
    />
```

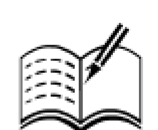
（3）打开 MainActivity.java 文件，在 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btnStart);        //获取 btnStart 组件
        btnButton.setOnClickListener(listener);                   //为 btnStart 设置监听事件
}
```

（4）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法，在该方法中使用 startActivity()方法启动一个 Activity。其代码如下：

```
private OnClickListener listener=new OnClickListener() {        //创建监听事件对象
    @Override
    public void onClick(View v) {
        //TODO Auto-generated method stub
        Intent intent=new Intent();                             //创建 Intent 对
```





Note

```
            //为 Intent 设置要打开的 Activity
            intent.setClass(MainActivity.this, TestActivity.class);
            startActivity(intent);                                    //通过 Intent 启动 Activity
        }
};
```

运行本实例，将显示如图 8.10 所示的运行效果。

单击图 8.10 中的“启动 Activity”按钮，跳转到 TestActivity 窗口，如图 8.11 所示。

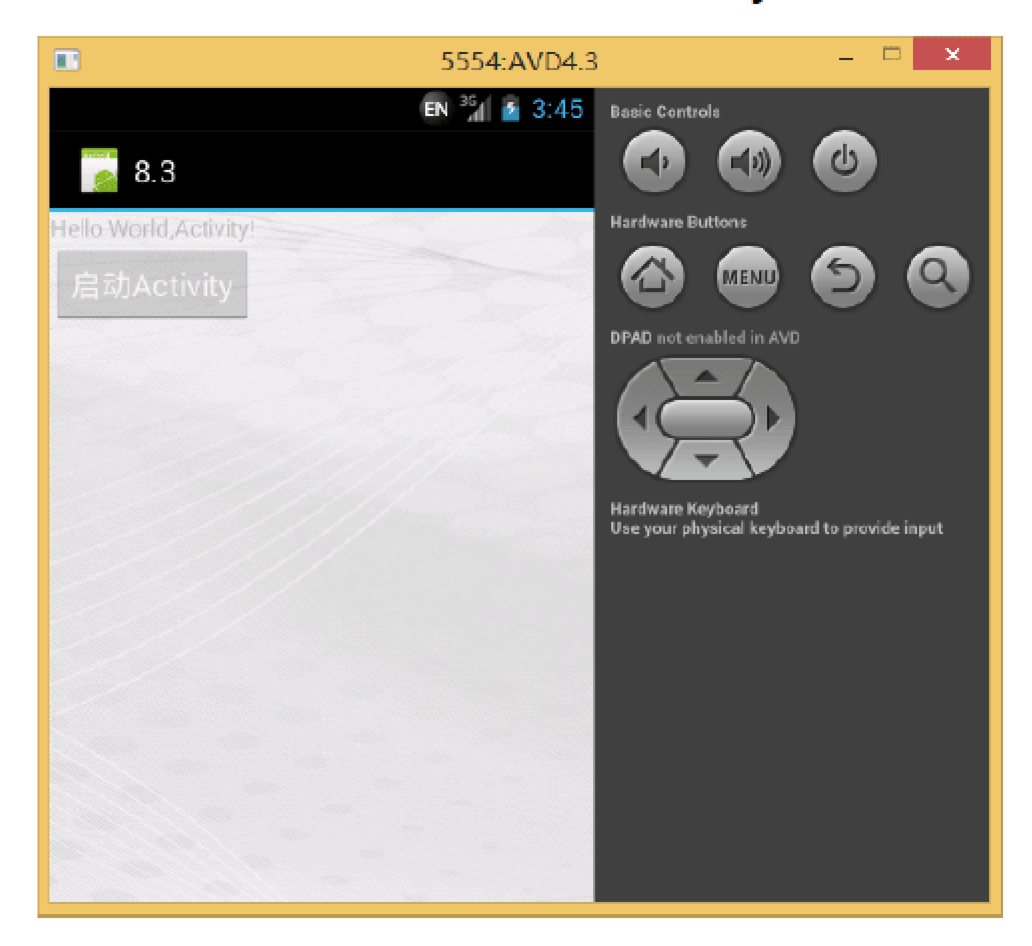


图 8.10　启动一个 Activity 程序的主 Activity

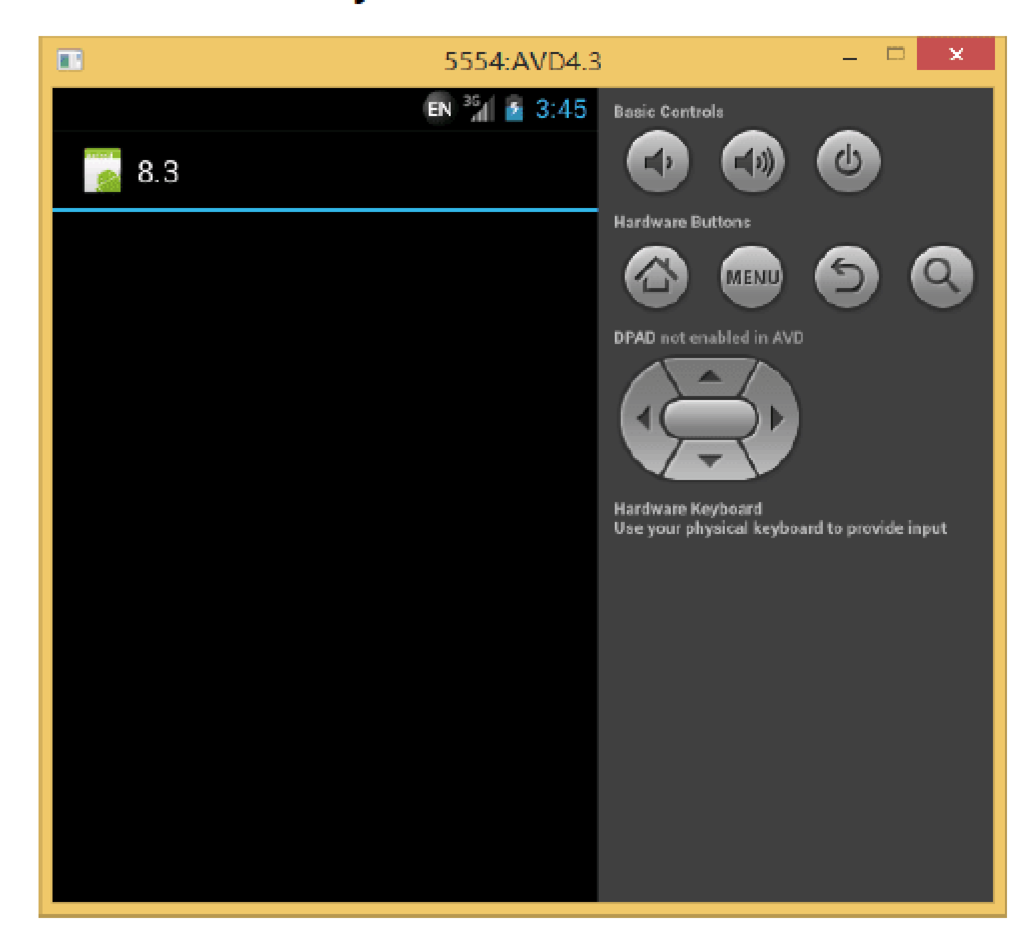


图 8.11　TestActivity 窗口

2．startActivities()方法

startActivities()方法用来同时启动多个 Activity，其语法格式如下：

```
public void startActivities(Intent[] intents)
```

参数 intents 表示要启动的多个 Intent 对象数组。

【例 8.4】 在 Eclipse 中创建 Android 项目，主要实现使用 startActivities()方法启动多个 Activity 的功能。

实例位置：光盘\MR\Instance\08\8.4

程序的开发步骤如下：

（1）按照 8.3.1 节中讲解的步骤创建 3 个 Activity，分别命名为 FirstActivity、SecondActivity 和 ThirdActivity，并分别在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btnStart，并设置其文本为“启动多个 Activity”。其代码如下：

```
<Button
    android:id="@+id/btnStart"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="启动多个 Activity"
    />
```


Note

（3）打开 MainActivity.java 文件，在 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btnStart);          //获取 btnStart 组件
        btnButton.setOnClickListener(listener);                      //为 btnStart 设置监听事件
}
```

（4）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，首先创建 3 个 Intent 对象，分别设置要打开的 3 个 Activity，然后创建 Intent 对象数组，该数组中存储刚才创建的 3 个 Intent 对象，最后使用 startActivities()方法同时启动 3 个 Activity。其主要代码如下：

```
private OnClickListener listener=new OnClickListener() {              //创建监听事件对象
    @Override
    public void onClick(View v) {
        //TODO Auto-generated method stub
        Intent intent1=new Intent();                                    //创建第一个 Intent 对象
        //为 Intent 设置要打开的 Activity
        intent1.setClass(MainActivity.this, FirstActivity.class);
        Intent intent2=new Intent();                                    //创建第二个 Intent 对象
        //为 Intent 设置要打开的 Activity
        intent2.setClass(MainActivity.this, SecondActivity.class);
        Intent intent3=new Intent();                                    //创建第三个 Intent 对象
        //为 Intent 设置要打开的 Activity
        intent3.setClass(MainActivity.this, ThirdActivity.class);
        Intent [] intents={intent1,intent2,intent3};                    //创建 Intent 数组
        startActivities(intents);                                       //通过 Intent 启动 Activity
    }
};
```

运行本实例，将显示如图 8.12 所示的运行效果。

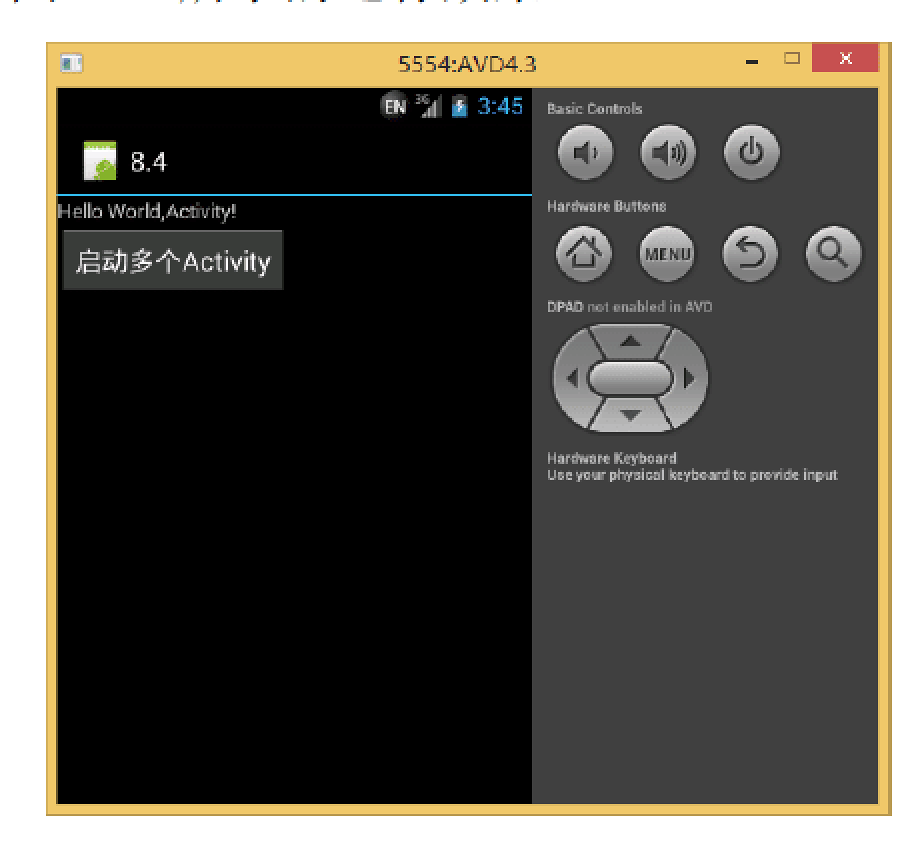
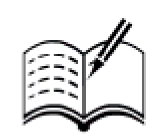


图 8.12　启动多个 Activity 程序的主 Activity



当用户单击图 8.12 中的“启动多个 Activity”按钮时，会同时启动 FirstActivity、SecondActivity 和 ThirdActivity，但是由于 Android 模拟器中同一时刻只能显示一个普通 Activity 窗口，所以展示给用户的是 ThirdActivity，而当用户单击模拟器中的返回按钮时，会依次显示之前启动的 SecondActivity 和 FirstActivity，在 LogCat 管理器中监测到的结果如图 8.13 所示。



L...	Time	PID	Application	Tag	Text
I	12-02 ...	783	com.xiaoke.e...	ACTIVITY	ThirdActivity==)onCreate
I	12-02 ...	783	com.xiaoke.e...	ACTIVITY	SecondActivity==)onCreate
I	12-02 ...	783	com.xiaoke.e...	ACTIVITY	FirstActivity==)onCreate

图 8.13 LogCat 管理器中监测的结果

8.3.3 多个 Activity 之间的传值

当程序中用到多个 Activity 时，常常需要涉及 Activity 间的传值问题，这时需要用到 Intent 对象的 putExtra()方法、getExtras()方法和 Bundle 对象。下面主要通过一个实例讲解如何在多个 Activity 之间实现相互传值。

【例 8.5】 在 Eclipse 中创建 Android 项目，主要实现在两个 Activity 间传值的功能。

实例位置：光盘\MR\Instance\08\8.5

程序的开发步骤如下：

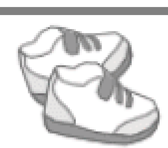
（1）按照 8.3.1 节中讲解的步骤创建一个 Activity，命名为 AcceptdataActivity，并在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btn，并设置其文本为“跳转”。其代码如下：

```
<Button
    android:id="@+id/btn"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:text="跳转"
    />
```

（3）在 res\layout 目录下创建一个 link.xml 文件，用来作为 AcceptdataActivity 的布局文件，在该布局文件中添加一个 TextView 组件和一个 Butotn 组件。其主要代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="链接页面"
```




Note

```
            />
            <Button
            android:id="@+id/btnBack"
            android:layout_width="60dp"
            android:layout_height="40dp"
            android:text="返回"
            />
</LinearLayout>
```

（4）打开 MainActivity.java 文件，定义一个 int 类型常量，用来作为请求标识。其代码如下：

```
private final static int REQUEST_CODE=1;                        //声明请求标识
```

（5）在 MainActivity.java 文件的 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);           //获取 Button 按钮
        btnButton.setOnClickListener(listener);                 //为 Button 按钮设置监听事件
}
```

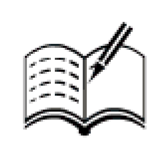
（6）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，首先创建一个 Intent 对象，并设置要打开的 Activity，然后使用 Intent 对象的 putExtra()方法设置要传递的值，最后使用 startActivityForResult()方法启动 Activity。其主要代码如下：

```
private OnClickListener listener=new OnClickListener() {           //创建监听对象
    @Override
    public void onClick(View v) {
        //TODO Auto-generated method stub
        Intent intent=new Intent();                                 //创建 Intent 对象
        //设置要访问的 Activity
        intent.setClass(MainActivity.this, AcceptdataActivity.class);
        intent.putExtra("str", "第一个 Activity 传过来的值");          //设置要传递的值
        startActivityForResult(intent, REQUEST_CODE);               //启动 Activity
    }
};
```

（7）在 MainActivity.java 文件中重写 onActivityResult()方法，该方法中主要实现获取 Activity 返回值的功能。其代码如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    //TODO Auto-generated method stub
    if(requestCode==REQUEST_CODE){                              //判断返回标识是否等于请求标识
        if(resultCode==AcceptdataActivity.RESULT_CODE){         //判断结果标识
```






```
                Bundle bundle=data.getExtras();             //获取返回值，并用 Bundle 接收
                String str=bundle.getString("back");        //获取 Bundle 中的返回值
                //弹出对话框，显示返回值
                Toast.makeText(MainActivity.this, str, Toast.LENGTH_LONG).show();
            }
        }
}
```

（8）打开 AcceptdataActivity.java 文件，首先定义 TextView 和 Button 对象，然后定义一个 int 类型常量，用来作为结果标识。其代码如下：

```
private TextView txt;                              //创建 TextView 组件对象
private Button btnButton;                          //创建 Button 组件对象
public final static int RESULT_CODE=1;             //定义结果标识
```

（9）在 AcceptdataActivity.java 文件的 onCreate()方法中，首先使用 Intent 对象的 getExtras()方法获取传递的值，并使用 Bundle 对象接收，然后将获取到的值设置为 TextView 组件的文本，最后获取布局文件中的 Button 按钮，并为其设置单击监听事件。OnCreate()方法的代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.link);
    Intent intent=getIntent();                         //创建 Intent 对象
    Bundle bundle=intent.getExtras();                  //获取传递值，并用 Bundle 接收
    String str=bundle.getString("str");                //获取传递的字符串值
    txt=(TextView)findViewById(R.id.txt);              //获取 TextView 组件
    txt.setText(str);                                  //设置文本
    btnButton=(Button)findViewById(R.id.btnBack);      //获取 Button 组件
    btnButton.setOnClickListener(listener);            //为 Button 设置监听事件
}
```

说明：

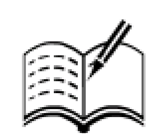
AcceptdataActivity.java 文件使用 link.xml 作为布局文件。

（10）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，首先创建一个 Intent 对象，并使用 Intent 对象的 putExtra()方法设置要返回的值，然后使用 setResult()方法设置返回标识，最后使用 finish()方法关闭当前 Activity。其主要代码如下：

```
private OnClickListener listener=new OnClickListener() {          //创建监听对象
    @Override
    public void onClick(View v) {
        //TODO Auto-generated method stub
        Intent intent=new Intent();                               //创建 Intent 对象
        intent.putExtra("back", "第二个 Activity 返回的值");       //设置返回值
        setResult(RESULT_CODE, intent);                           //设置返回标识
```



```
            finish();                                   //关闭 Activity
        }
};
```



Note

运行本实例，将显示如图 8.14 所示的运行效果。单击“跳转”按钮，进入第二个 Activity，该 Activity 窗口中显示第一个 Activity 传过来的值，如图 8.15 所示；单击“返回”按钮，返回到第一个 Activity，该 Activity 窗口中弹出提示框，显示第二个 Activity 返回的值，如图 8.16 所示。

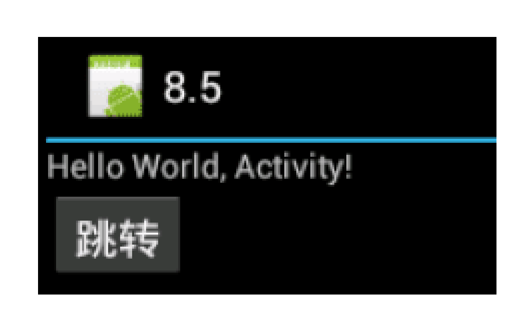


图 8.14　主 Activity 初始页面

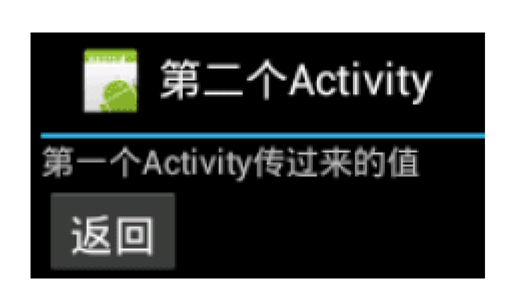


图 8.15　第二个 Activity 接收的值



图 8.16　接收第二个 Activity 的返回值

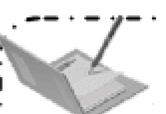
8.3.4　关闭 Activity

关闭 Activity 使用 finish()方法实现，该方法用来关闭当前 Activity，其语法格式如下：

```
public void finish()
```

在 Android 程序中，如果要关闭当前的 Activity，直接使用下面语句即可。

```
finish();
```



说明：

关闭 Activity 还可以使用 finishActivity()方法实现，该方法用来关闭使用 startActivityForResult()方法启动的 Activity，该方法的语法中有一个 int 类型的参数，用来表示 Activity 的请求标识。

8.4　综 合 应 用

8.4.1　根据输入的生日判断星座

【例 8.6】 在占星学上，黄道十二星座是宇宙方位的代名词，十二星座代表了 12 种基本性





格原型，一个人出生时各星体落入黄道上的位置，正是说明着一个人的先天性格及天赋。因此，现在很多人都希望知道自己的星座。本实例将实现根据输入的阳历生日判断所属星座。运行程序，将显示一个输入阳历生日的界面，输入正确的生日后，如图 8.17 所示，单击"确定"按钮，将显示如图 8.18 所示的判断结果界面。

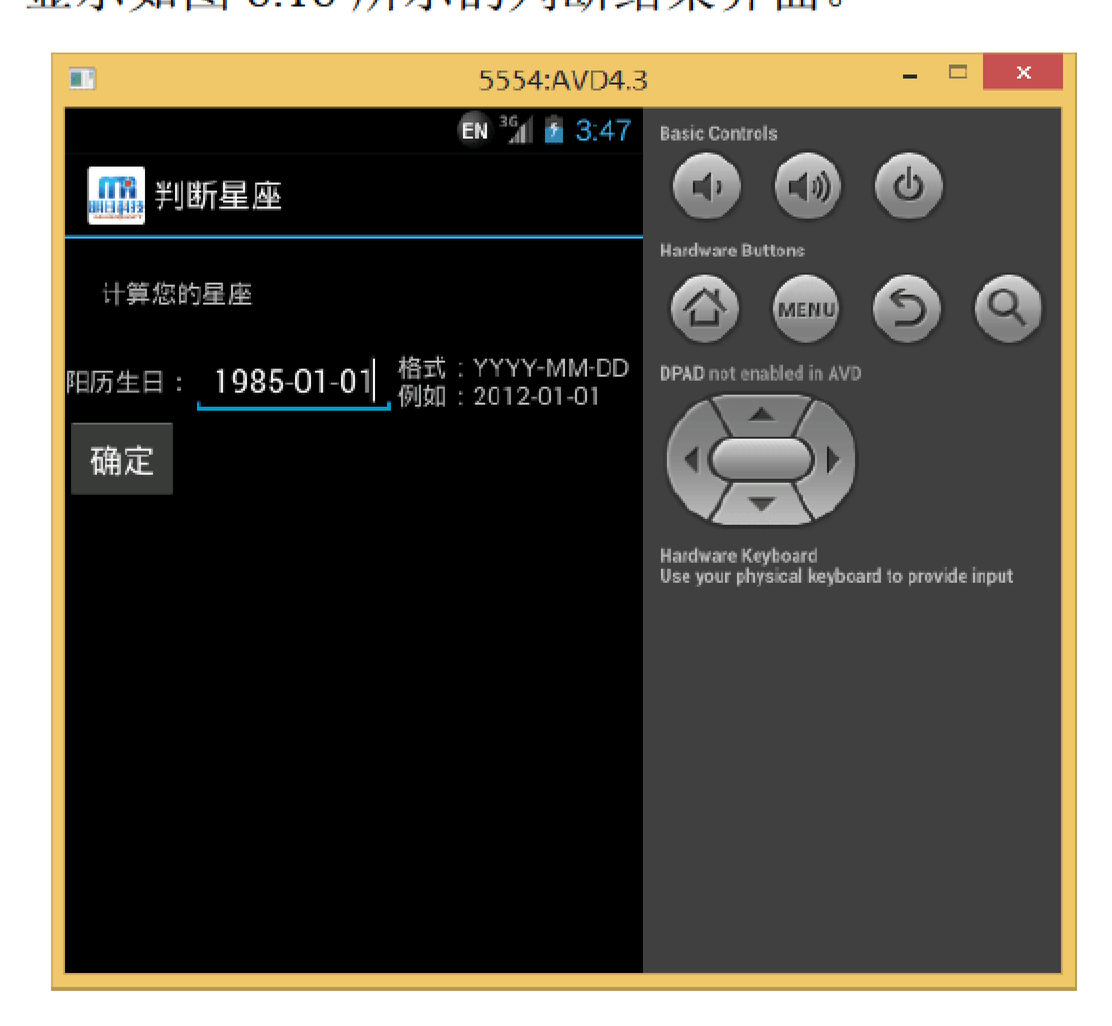


图 8.17　输入阳历生日的界面

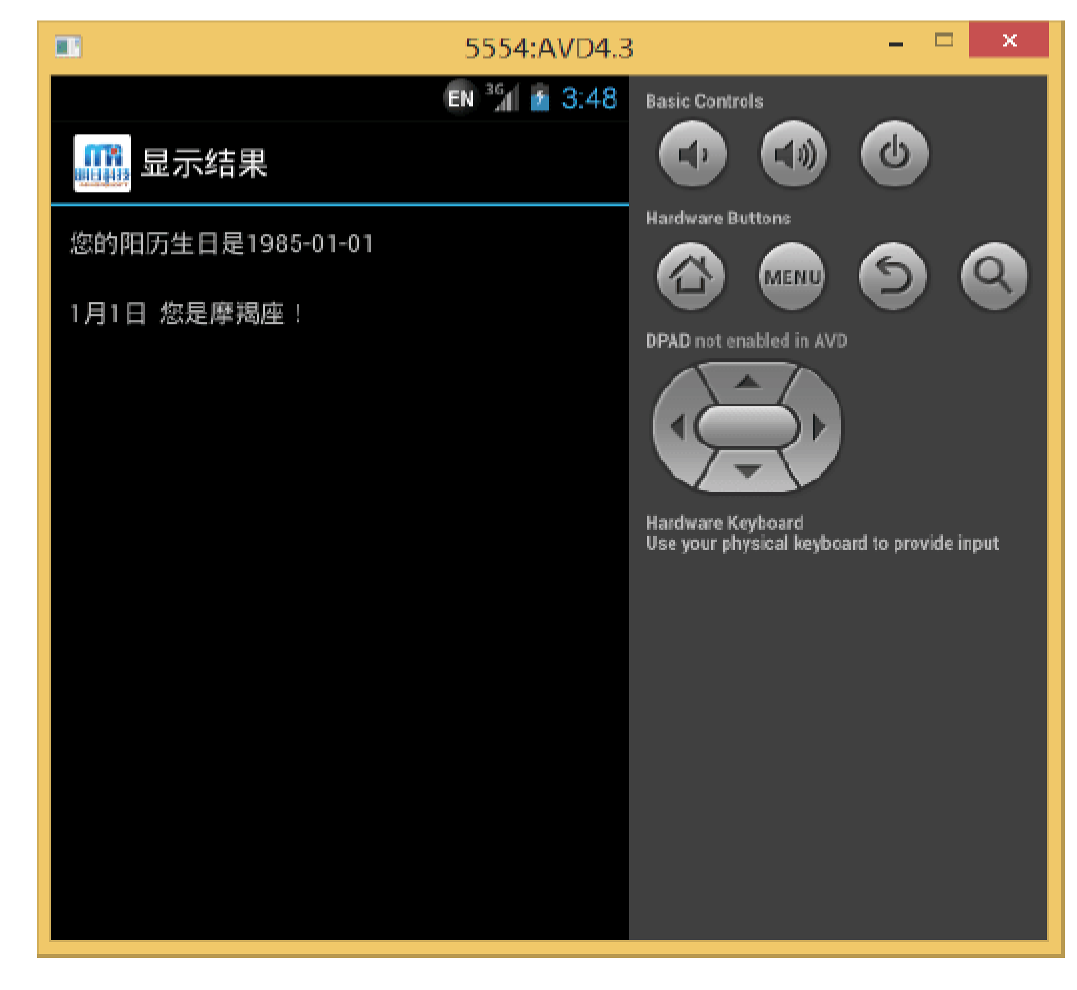


图 8.18　显示判断结果的界面

实例位置：光盘\MR\Instance\08\8.6

程序的开发步骤如下：

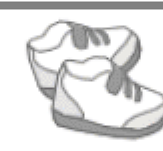
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认添加的垂直线性布局管理器中，添加用于输入生日的编辑框和"确定"按钮，以及一些用于显示说明信息的文本框。

（2）编写一个实现 java.io.Serializable 接口的 Java 类，在该类中，创建一个用于保存生日的属性，并为该属性添加对应的 setter()和 getter()方法。其关键代码如下：

```
public class Info implements Serializable {
	private static final long serialVersionUID = 1L;
	private String birthday="";			//生日
	public String getBirthday() {
		return birthday;
	}
	public void setBirthday(String birthday) {
		this.birthday = birthday;
	}
}
```

说明：

在使用 Bundle 类传递数据包时，可以放入一个可序列化的对象。这样，当要传递的数据字段比较多时，采用该方法比较方便。在实现本实例时，为了在 Bundle 中放入一个可序列化的对象，所以创建了一个可序列化的 Java 类，方便存储可序列化对象。

Note

（3）打开默认创建的主活动，也就是 MainActivity，在 onCreate()方法中，获取“确定”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，实例化一个保存生日的可序列化对象 info，并判断是否输入生日，如果没有输入，则给出消息提示，并返回，否则，首先获取生日并保存到 info 中，然后实例化一个 Bundle 对象，并将输入的生日保存到 Bundle 对象中，接下来再创建一个启动显示结果 Activity 的 intent 对象，并将 Bundle 对象保存到该 intent 对象中，最后启动 intent 对应的 Activity。其关键代码如下：

```
Button button=(Button)findViewById(R.id.button1);
button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            Info info=new Info();                              //实例化一个保存输入基本信息的对象
            if("".equals(((EditText)findViewById(R.id.birthday)).getText().toString())){
              Toast.makeText(MainActivity.this, "请输入您的阳历生日，否则不能计算！", Toast.LENGTH_
SHORT).show();
              return;
            }
            String birthday=((EditText)findViewById(R.id.birthday)).getText().toString();

            info.setBirthday(birthday);                        //设置生日
            Bundle bundle=new Bundle();                        //实例化一个 Bundle 对象
            bundle.putSerializable("info", info);              //将输入的基本信息保存到 Bundle 对象中
            Intent intent=new Intent(MainActivity.this,ResultActivity.class);
            intent.putExtras(bundle);                          //将 bundle 保存到 Intent 对象中
            startActivity(intent);                             //启动 intent 对应的 Activity
        }
});
```

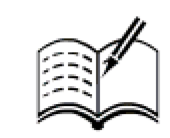
说明：

在上面的代码中，加粗的代码用于创建一个 Bundle 对象，并在该对象中放入一个可序列化的 Info 类的对象。

（4）在 res\layout 目录中，创建一个名称为 result.xml 的布局文件，在该布局文件中采用垂直线性布局，并且添加两个 TextView 组件，分别用于显示生日和计算结果。result.xml 的具体代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical" >
      <TextView
          android:id="@+id/birthday"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:padding="10px"
          android:text="阳历生日" />
```





Note

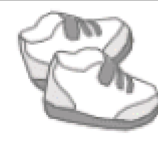
```
    <TextView
        android:id="@+id/result"
        android:padding="10px"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="星座" />
</LinearLayout>
```

（5）在 com.mingrisoft 包中，创建一个继承 Activity 类的 ResultActivity，并且重写 onCreate()方法。在重写的 onCreate()方法中，首先设置该 Activity 使用的布局文件 result.xml 中定义的布局，然后获取生日和显示结果文本框，再获取 Intent 对象，以及传递的数据包，最后将传递过来的生日和判断结果显示到对应的文本框中。其关键代码如下：

```
setContentView(R.layout.result);                                    //设置该 Activity 使用的布局
TextView birthday = (TextView) findViewById(R.id.birthday);        //获取显示生日的文本框
TextView result = (TextView) findViewById(R.id.result);            //获取显示星座的文本框
Intent intent = getIntent();                                        //获取 Intent 对象
Bundle bundle = intent.getExtras();                                 //获取传递的数据包
Info info = (Info) bundle.getSerializable("info");                 //获取一个可序列化的 info 对象
birthday.setText("您的阳历生日是" + info.getBirthday());           //获取性别并显示到相应文本框中
result.setText( query(info.getBirthday()));                         //显示计算后的星座
```

（6）编写根据阳历生日判断星座的方法 query()，该方法包括一个入口参数，用于指定生日，返回值为字符串类型的所属星座。query()方法的具体代码如下：

```
/**
 * 功能根据生日查询星座
 *
 * @param birthday
 * @return
 */
public String query(String birthday) {
    int month=0;                                                    //月
    int day=0;                                                      //日
    try{                                                            //捕获异常
        month=Integer.parseInt(birthday.substring(5, 7));           //获取输入的月份
        day=Integer.parseInt(birthday.substring(8, 10));            //获取输入的日
    }catch(Exception e){
        e.printStackTrace();
    }
    String name = "";                                               //提示信息
    if (month > 0 && month < 13 && day > 0 && day < 32) {          //如果输入的月和日有效
        if ((month == 3 && day > 20) || (month == 4 && day < 21)) {
            name = "您是白羊座！";
        } else if ((month == 4 && day > 20) || (month == 5 && day < 21)) {
            name = "您是金牛座！";
        } else if ((month == 5 && day > 20) || (month == 6 && day < 22)) {
            name = "您是双子座！";
        } else if ((month == 6 && day > 21) || (month == 7 && day < 23)) {
```

Note

```
                name = "您是巨蟹座！";
            } else if ((month == 7 && day > 22) || (month == 8 && day < 23)) {
                name = "您是狮子座！";
            } else if ((month == 8 && day > 22) || (month == 9 && day < 23)) {
                name = "您是处女座！";
            } else if ((month == 9 && day > 22) || (month == 10 && day < 23)) {
                name = "您是天平座！";
            } else if ((month == 10 && day > 22) || (month == 11 && day < 22)) {
                name = "您是天蝎座！";
            } else if ((month == 11 && day > 21) || (month == 12 && day < 22)) {
                name = "您是射手座！";
            } else if ((month == 12 && day > 21) || (month == 1 && day < 20)) {
                name = "您是摩羯座！";
            } else if ((month == 1 && day > 19) || (month == 2 && day < 19)) {
                name = "您是水牛座！";
            } else if ((month == 2 && day > 18) || (month == 3 && day < 21)) {
                name = "您是双鱼座！";
            }
            name = month + "月" + day + "日  " + name;
        } else {                                        //如果输入的月和日无效
            name = "您输入的生日格式不正确或者不是真实生日！";
        }
        return name;                                    //返回星座或提示信息
}
```

（7）在 AndroidManifest.xml 文件中配置 ResultActivity，配置的主要属性有 Activity 使用的标签、图标和实现类。其具体代码如下：

```
<activity
    android:label="显示结果"
    android:icon="@drawable/ic_launcher"
    android:name=".ResultActivity">
</activity>
```

8.4.2 带选择头像的用户注册界面

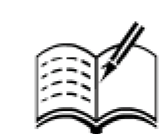
【例 8.7】 本实例主要实现带选择头像的用户注册界面，打开新的 Activity 选择头像，并将选择的头像返回到原 Activity 中。运行程序，将显示一个填写用户注册信息的界面，输入用户名、密码、确认密码和 E-mail 地址后，单击“选择头像”按钮，将打开如图 8.19 所示的选择头像的界面，单击想要的头像，将返回到填写用户注册信息界面，如图 8.20 所示。

实例位置：光盘\MR\Instance\08\8.7

程序的开发步骤如下：

（1）在 main.xml 布局文件中，将默认添加的垂直线性布局管理器修改为水平线性布局管理器，并将默认添加的 TextView 组件删除，然后添加两个垂直线性布局管理器，并在第一个线性





布局管理器中添加一个 4 行的表格布局管理器，在第二个线性布局管理器中添加一个 ImageView 组件和一个 Button 组件，最后在表格布局管理器的各行中添加用于输入用户名、密码和 E-mail 地址等的 TextView 和 EditText 组件。

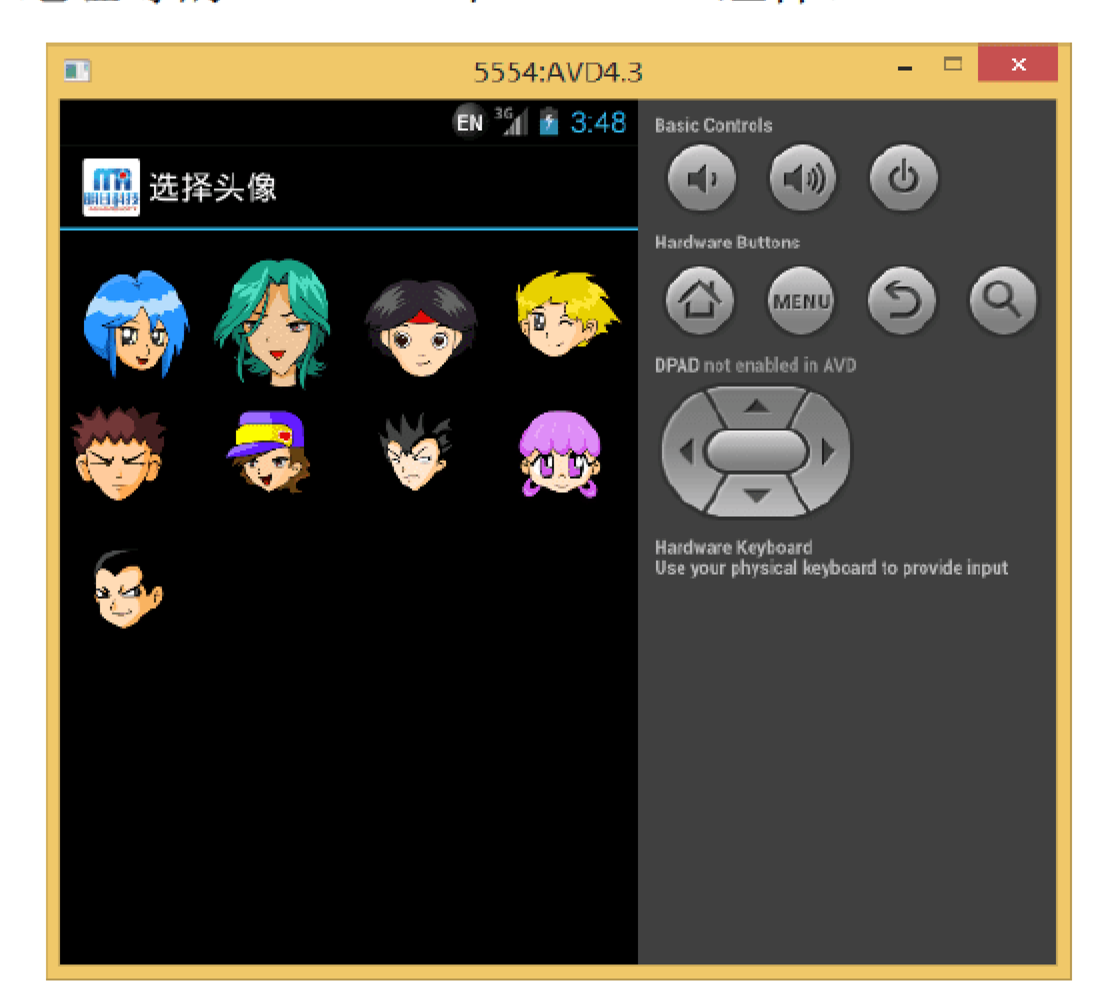


图 8.19　选择头像界面



图 8.20　填写用户注册信息界面

（2）打开默认创建的主活动，也就是 MainActivity，在 onCreate()方法中获取“选择头像”按钮，并为其添加单击事件监听器，在重写的 onClick()方法中创建一个要启动的 Activity 对应的 Intent 对象，并应用 startActivityForResult()方法启动指定的 Activity 并等待返回结果。其具体代码如下：

```
Button button=(Button)findViewById(R.id.button1);          //获取“选择头像”按钮
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent(MainActivity.this,HeadActivity.class);
        startActivityForResult(intent, 0x11);              //启动指定的 Activity
    }
});
```

（3）在 res\layout 目录中，创建一个名称为 head.xml 的布局文件，在该布局文件中采用垂直线性布局，并且添加一个 GridView 组件，用于显示可选择的头像列表。其关键代码如下：

```
<GridView android:id="@+id/gridView1"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:layout_marginTop="10px"
    android:horizontalSpacing="3px"
    android:verticalSpacing="3px"
    android:numColumns="4"
/>
```

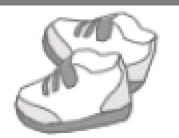
（4）在 com.mingrisoft 包中，创建一个继承 Activity 类的 HeadActivity，并且重写 onCreate()

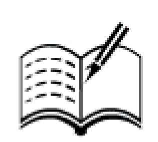
方法，然后定义一个保存要显示头像 id 的一维数组。其关键代码如下：

```
public int[] imageId = new int[] { R.drawable.img01, R.drawable.img02,
        R.drawable.img03, R.drawable.img04, R.drawable.img05,
        R.drawable.img06, R.drawable.img07, R.drawable.img08,
        R.drawable.img09
};                                                              //定义并初始化保存头像 id 的数组
```

（5）在重写的 onCreate()方法中，首先设置该 Activity 使用布局文件 head.xml 中定义的布局，然后获取 GridView 组件，并创建一个与之关联的 BaseAdapter 适配器。其关键代码如下：

```
setContentView(R.layout.head);                                  //设置该 Activity 使用的布局
GridView gridview = (GridView) findViewById(R.id.gridView1);    //获取 GridView 组件
BaseAdapter adapter=new BaseAdapter() {
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
                ImageView imageview;                            //声明 ImageView 的对象
                if(convertView==null){
                        imageview=new ImageView(HeadActivity.this);     //实例化 ImageView 的对象
                        /*************设置图像的宽度和高度*****************/
                        imageview.setAdjustViewBounds(true);
                        imageview.setMaxWidth(158);
                        imageview.setMaxHeight(150);
                        /**************************************************/
                        imageview.setPadding(5, 5, 5, 5);       //设置 ImageView 的内边距
                }else{
                        imageview=(ImageView)convertView;
                }
                imageview.setImageResource(imageId[position]);  //为 ImageView 设置显示的图片
                return imageview;                               //返回 ImageView
        }
        /*
         * 功能：获得当前选项的 id
         */
        @Override
        public long getItemId(int position) {
                return position;
        }
        /*
         * 功能：获得当前选项
         */
        @Override
        public Object getItem(int position) {
                return position;
        }
        /*
         * 获得数量
         */
        @Override
        public int getCount() {
```






```
            return imageId.length;
        }
};
gridview.setAdapter(adapter);                                   //将适配器与 GridView 关联
```

（6）为 GridView 添加 OnItemClickListener 事件监听器，在重写的 onItemClick()方法中，首先获取 Intent 对象，然后创建一个要传递的数据包，并将选中的头像 id 保存到该数据包中，再将要传递的数据包保存到 intent 中，并设置返回的结果码，及返回的 Activity，最后关闭当前 Activity。其关键代码如下：

```
gridview.setOnItemClickListener(new OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,long id) {
            Intent intent=getIntent();                              //获取 Intent 对象
            Bundle bundle=new Bundle();                             //实例化传递的数据包
            bundle.putInt("imageId",imageId[position] );            //显示选中的图片
            intent.putExtras(bundle);                               //将数据包保存到 intent 中
            setResult(0x11,intent);                //设置返回的结果码，并返回调用该 Activity 的 Activity
            finish();                                               //关闭当前 Activity
        }
});
```

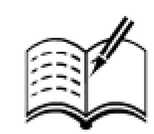
（7）重新打开 MainActivity，在该类中重写 onActivityResult()方法。在该方法中，需要判断 requestCode 请求码和 resultCode 结果码是否与预先设置的相同，如果相同，则获取传递的数据包，并从该数据包中获取出选择的头像 id，且显示选择的头像。其具体代码如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode==0x11 && resultCode==0x11){                  //判断是否为待处理的结果
            Bundle bundle=data.getExtras();                         //获取传递的数据包
            int imageId=bundle.getInt("imageId");                   //获取选择的头像 id
            //获取布局文件中添加的 ImageView 组件
            ImageView iv=(ImageView)findViewById(R.id.imageView1);
            iv.setImageResource(imageId);                           //显示选择的头像
        }
}
```

（8）在 AndroidManifest.xml 文件中配置 HeadActivity，配置的主要属性有 Activity 使用的标签、图标和实现类。其具体代码如下：

```
<activity
      android:label="选择头像"
      android:icon="@drawable/ic_launcher"
      android:name=".HeadActivity">
</activity>
```





Note

8.4.3 仿 QQ 客户端登录界面

【例 8.8】 本实例主要实现在第一个 Activity 中显示登录界面，输入正确的账号和密码后，启动另一个 Activity 显示当前登录用户的昵称。运行程序，在屏幕上将显示一个登录对话框，输入账号和密码后，如图 8.21 所示。单击“登录”按钮，将判断输入的账号和密码是否正确，如果正确，将打开如图 8.22 所示的主界面。在该界面中，将显示当前登录用户的昵称和“退出登录”按钮，单击“退出登录”按钮，将返回到如图 8.21 所示的用户登录界面。





图 8.21 登录界面

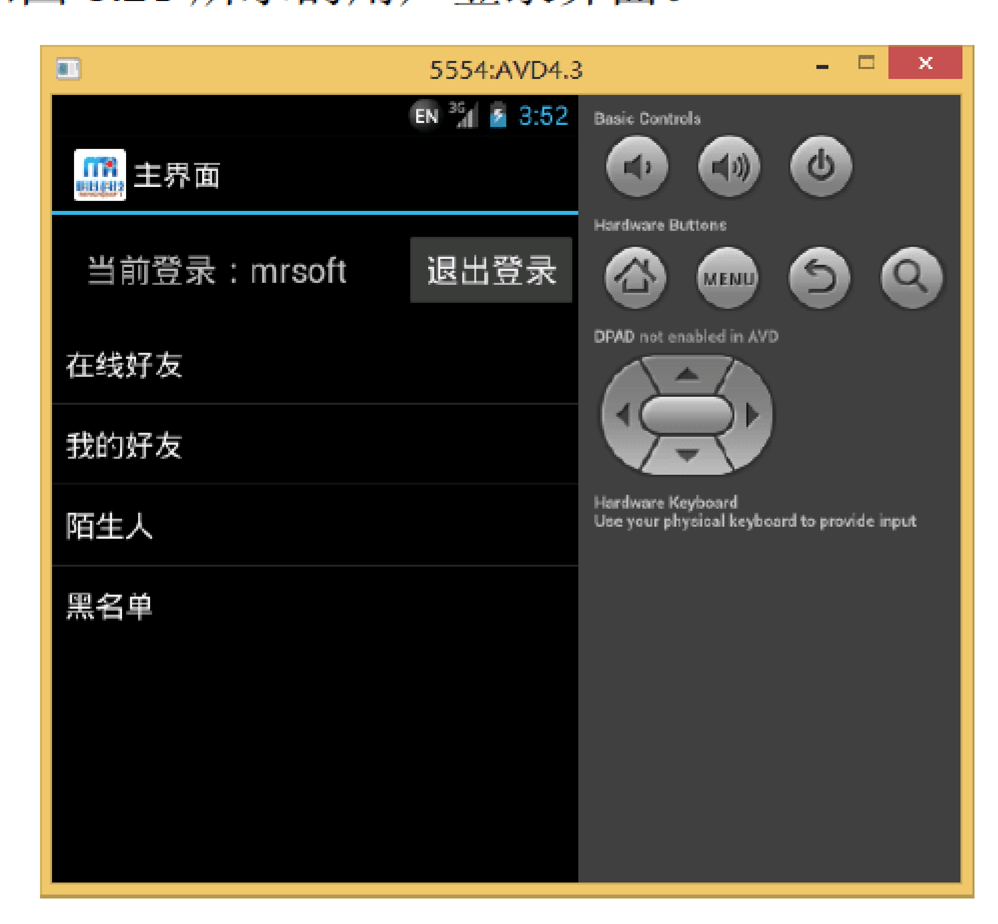


图 8.22 显示昵称的主界面

实例位置：光盘\MR\Instance\08\8.8

程序的开发步骤如下：

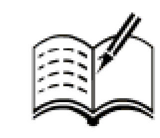
（1）在 res\layout 目录下创建布局文件 login.xml，在该文件中应用表格布局完成用户登录界面，包括用于输入登录账号的编辑框和输入密码的编辑框。

（2）在 com.mingrisoft 包中，创建一个 final 类，在该中创建一个保存用户信息的常量数组。其具体代码如下：

```
public final class Data {
    //用户信息
    public static final String[][] USER = {
     {"1001","111","明日"},
     {"1002","111","mrsoft"},
     {"1003","111","wgh"}
    };
}
```

（3）在 com.mingrisoft 包中，创建一个继承 android.app.Activity 的 LoginActivity，并重写 onCreate()方法。在重写的 onCreate()方法中，首先获取“登录”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，获取输入的账号和密码，并判断账号和密码是否正确，如果正确将对应的昵称保存到 Intent 中，并启动主界面 MainActivity，然后获取“退出”按钮，并为其



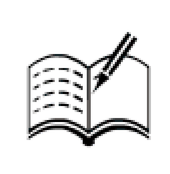


添加单击事件监听器，最后应用 finish()方法，关闭当前 Activity。其关键代码如下：

```
public class LoginActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login);                    //设置该 Activity 使用的布局
        Button button=(Button)findViewById(R.id.login);
        button.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                String number=((EditText)findViewById(R.id.editText1)).getText().toString();
                String pwd=((EditText)findViewById(R.id.editText2)).getText().toString();
                boolean flag=false;                        //用于记录登录是否成功的标记变量
                String nickname="";                        //保存昵称的变量
                //通过遍历数据的形式判断输入的账号和密码是否正确
                for(int i=0;i<Data.USER.length;i++){
                    if(number.equals(Data.USER[i][0])){    //判断账号是否正确
                        if(pwd.equals(Data.USER[i][1])){   //判断密码是否正确
                            nickname=Data.USER[i][2];      //获取昵称
                            flag=true;                     //将标志变量设置为 true
                            break;                         //跳出 for 循环
                        }
                    }
                }
                if(flag){
                    //创建要显示 Activity 对应的 Intent 对象
                    Intent intent=new Intent(LoginActivity.this,MainActivity.class);
                    Bundle bundle=new Bundle();            //创建一个 Bundle 的对象 bundle
                    bundle.putString("nickname", nickname); //保存昵称
                    intent.putExtras(bundle);              //将数据包添加到 intent 对象中
                    startActivity(intent);                 //开启一个新的 Activity
                }else{
                    Toast.makeText(LoginActivity.this,
                            "您输入的账号或密码错误！", Toast.LENGTH_SHORT);
                }
            }
        });
        Button exit=(Button)findViewById(R.id.exit);
        exit.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();                                  //关闭当前 Activity
            }
        });
    }
}
```

（4）打开默认创建的 main.xml 文件，将默认添加的 TextView 组件删除，然后添加一个水平的线性布局管理器和一个 ListView 组件，并且在线性布局管理器中，再添加一个 id 为 nickname





Note

的 TextView 组件和一个 id 为 m_exit 的 Button 组件。其关键代码如下：

```
<LinearLayout
        android:id="@+id/linearLayout2"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
              android:id="@+id/nickname"
              android:layout_width="wrap_content"
              android:layout_weight="9"
              android:textSize="20px"
              android:padding="20px"
              android:layout_height="wrap_content"
              android:text="TextView" />
        <Button
              android:id="@+id/m_exit"
              android:layout_weight="1"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
android:textSize="20px"
              android:text="退出登录" />
</LinearLayout>
<ListView
        android:id="@+id/listView1"
        android:entries="@array/option"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
</ListView>
```

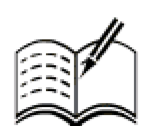
说明：

上面的代码中，加粗的代码用于通过数组资源为 ListView 组件设置要显示的列表项。所以还需要在 res\value 目录中创建一个定义数组资源的 XML 文件 arrays.xml，并在该文件中添加名称为 option 的字符串数组。其关键代码如下：

```
<resources>
      <string-array name="option">
        <item>在线好友</item>
        <item>我的好友</item>
        <item>陌生人</item>
        <item>黑名单</item>
      </string-array>
</resources>
```

（5）打开默认添加的 MainActivity，在 onCreate()方法中，首先获取 Intent 对象以及传递的数据包，然后通过该数据包获取传递的昵称，再获取显示登录用户的 TextView 组件，并通过该





组件显示登录用户的昵称，最后获取“退出登录”按钮，并为其添加单击事件监听器，在重写的 onClick()方法中，关闭当前 Activity。其关键代码如下：

```
Intent intent=getIntent();                              //获取 Intent 对象
Bundle bundle=intent.getExtras();                       //获取传递的数据包
String nickname=bundle.getString("nickname");           //获取传递过来的昵称
TextView tv=(TextView)findViewById(R.id.nickname);//获取用于显示当前登录用户的 TextView 组件
tv.setText("当前登录："+nickname);                        //显示当前登录用户的昵称
Button button=(Button)findViewById(R.id.m_exit);        //获取“退出登录”按钮
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();                                       //关闭当前 Activity
    }
});
```

（6）打开 AndroidManifest.xml 文件，修改默认的配置代码。在该文件中，首先修改入口 Activity，这里修改为 LoginActivity，并为其设置 android:theme 属性，然后配置 MainActivity。修改后的关键代码如下：

```
<activity
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Dialog"
    android:name=".LoginActivity" >
    <intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
     android:name=".MainActivity"
     android:label="主界面" />
```

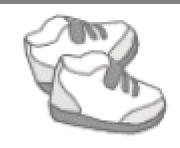
8.5 本章常见错误

运行 Android 程序时，出现下面的错误提示。

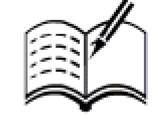
```
android.content.ActivityNotFoundException:Unable to find explicit activity class
```

出现该错误主要是由于 Activity 没有在 AndroidManifest.xml 文件中声明引起的，解决该错误时，只要在 AndroidManifest.xml 文件中添加相应 Activity 的声明即可，例如：

```
<activity
    android:name=".Activity 名称"
    <intent-filter>
```

```
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```



Note

8.6 本章小结

本章主要对 Android 程序的核心对象——Activity 进行了详细讲解，其中，主要包括 Activity 的概述、4 种状态、常用属性及方法、生命周期及其常用操作（如创建、启动、传值、关闭等）。Activity 是 Android 程序中最基本、也是最核心的模块，它为用户操作提供了方便快捷的可视化用户界面，所以在学习本章内容时，读者一定要熟练掌握本章所讲解的所有内容，并重点掌握 Activity 的常见操作，将其应用于 Android 实际项目开发中。

8.7 跟我上机

参考答案：光盘\MR\跟我上机

在玩手机游戏时，经常会看到“关于”按钮，单击该按钮可以查看关于该游戏的介绍，这里将开发一个 Android 程序，实现一个泡泡龙游戏的关于功能，即在游戏的主窗体上放置一个“关于”按钮。单击该按钮，启动一个对话框主题的 Activity，用于显示游戏的关于信息。具体实现时，首先需要在布局文件 main.xml 中应用线性布局和相对布局完成一个带“关于”按钮的游戏开始界面。然后，创建一个继承 Activity 类的 AboutActivity，并且重写 onCreate()方法，在重写的 onCreate()方法中，首先创建一个线性布局管理器对象，并设置其内边距，然后创建一个 TextView 对象，并设置字体大小及要显示的内容（泡泡龙游戏是一款十分流行的益智游戏。它可以从下方中央的弹珠发射台射出彩珠，当有多于 3 个同色弹珠相连时，这些弹珠将会爆掉，否则该弹珠被连接到指向的位置，直到泡泡下压越过下方的警戒线，游戏结束），再将 TextView 添加到线性布局管理器中，最后设置在该 Activity 中显示线性布局管理器对象。其参考代码如下：

```
public class AboutActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout ll=new LinearLayout(this);          //创建线性布局管理器对象
        ll.setPadding(10,10,10,10);
        TextView tv=new TextView(this);                  //创建 TextView 对象
        tv.setTextSize(18);                              //设置字体大小
        tv.setText(R.string.about);                      //设置要显示的内容
        ll.addView(tv);                                  //将 TextView 添加到线性布局管理器中
        setContentView(ll);                              //设置该 Activity 显示的内容视图
```

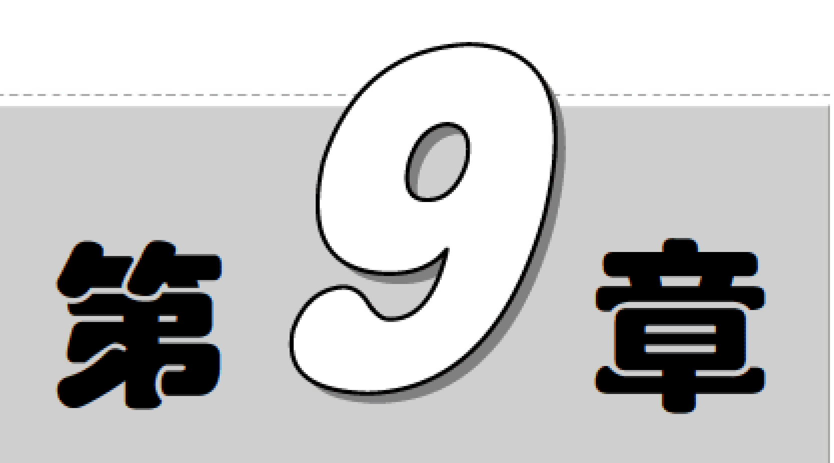


```
    }
}
```

打开默认创建的主活动，也就是 MainActivity，在 onCreate()方法中获取“关于”按钮，并为其添加单击事件监听器，在重写的 onClick()方法中创建一个 AboutActivity 所对应的 Intent 对象，并调用 startActivity()方法，启动 AboutActivity。其参考代码如下：

```
ImageView about=(ImageView)findViewById(R.id.about);                 //获取“关于”按钮
about.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建 Intent 对象
        Intent intent=new Intent(MainActivity.this, AboutActivity.class);
        startActivity(intent);                                        //启动关于 Activity
    }
});
```

Note



使用 Intent 进行通信

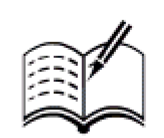
（视频讲解：56 分钟）

在 Android 程序中，其 3 大核心组件——Activity、Service 和 BroadcastReceiver 都是通过 Intent 消息来进行激活的。Intent 消息是一种同一或不同应用程序中的组件之间延迟运行时绑定的机制，它本身是一个对象，是一个被动的数据结构保存一个将要执行的操作的抽象描述。本章将对 Intent 对象的概述、组成、解析，以及如何使用该对象进行通信进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 通过 Intent 实现拨打电话和发送短信
- ☐ 为 Intent 添加附加信息和读取附加信息
- ☐ 将字符串数据传递到打开的 Activity 中
- ☐ 得到新打开 Activity 关闭后返回数据
- ☐ 使用 Intent 实现直接发送短信
- ☐ 使用 Intent 打开网页
- ☐ 使用 Intent 实现返回系统 Home 桌面





9.1 Intent 对象简介

Intent 是 Android 程序中传输数据的核心对象，在 Android 官方文档中，对 Intent 的定义是执行某操作的一个抽象描述。本节将对 Intent 对象及其常见的 3 种传输机制进行介绍。

9.1.1 Intent 对象概述

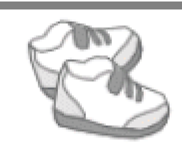
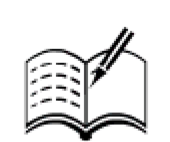
在一个 Android 程序中，主要是由 3 种组件组成的，这 3 种组件是独立的，它们之间可以互相调用、协调工作，最终组成一个真正的 Android 程序。在这些组件之间的通信中，主要是由 Intent 协助完成的。Intent 负责对应用中一次操作的动作、动作涉及数据和附加数据进行描述，Android 则根据此 Intent 的描述，负责找到对应的组件，将 Intent 传递给调用的组件，并完成组件的调用，因此，Intent 在这里起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。

例如，在一个联系人维护的应用中，当在一个联系人列表屏幕（假设对应的 Activity 为 listActivity）上单击某个联系人后，希望能够跳出此联系人的详细信息屏幕（假设对应的 Activity 为 detailActivity），为了实现这个目的，listActivity 需要构造一个 Intent，这个 Intent 用于告诉系统：要做"查看"动作，而此动作对应的查看对象是"某联系人"；然后调用 startActivity(Intent intent)方法将构造的 Intent 传入，系统会根据此 Intent 中的描述，在 AndroidManifest.xml 文件中找到满足此 Intent 要求的 Activity，即 detailActivity，并将其传入 Intent；最后，detailActivity 会根据此 Intent 中的描述，执行相应的操作。

9.1.2 3 种不同的 Intent 传输机制

Intent 对象主要用来在 Android 程序的 Activity、Service 和 BroadcastReceiver 这 3 大组件之间传输数据，而针对这 3 大组件，有独立的 Intent 传输机制，分别如下。

- ☑ Activity：通过将一个 Intent 对象传递给 Context.startActivity()或 Activity.startActivityForRestult()方法，启动一个活动或者使一个已存在的活动去做新的事情。
- ☑ Service：通过将一个 Intent 对象传递给 Context.startService()方法，初始化一个 Service 或者传递一个新的指令给正在运行的 Service；类似的，通过将一个 Intent 对象传递给 Context.bindService()方法，可以建立调用组件和目标服务之间的连接。
- ☑ BroadcastReceiver：通过将一个 Intent 对象传递给任何广播方法（如 Context.sendBroadcast()、Context.sendOrderedBroadcast()和 Context.sendStickyBroadcast()方法等），都可以传递到所有感兴趣的广播接收者。

Note

说明：

在每种传输机制下，Android 程序会自动查找合适的 Activity、Service 或者 BroadcastReceiver 来响应 Intent（意图），如果有必要的话，初始化它们，这些消息系统之间没有重叠，即广播意图只会传递给广播接收者，而不会传递活动或服务，反之亦然。

9.2 Intent 对象的组成

一个 Intent 对象实质上是一个捆绑信息，包含对 Intent 有兴趣的组件的信息（如要执行的动作和要作用的数据）、Android 系统有兴趣的信息（如处理 Intent 组件的分类信息和如何启动目标活动的指令等）。本节将对组成 Intent 对象的主要信息进行讲解。

9.2.1 组件名称

组件名称用来指定为处理 Intent 对象的组件，它是一个 ComponentName 对象，是目标组件的完全限定类名（如 com.xiaoke.project.app.IntentExamActivity）和应用程序所在的包在清单文件中的名字（如 com.xiaoke.project）的组合，其中组件名称中的包部分不必一定和清单文件中的包名一样。

组件名称是可选的，如果设置了，Intent 对象传递到指定类的实例；如果没有设置，Android 使用 Intent 中的其他信息来定位合适的目标组件。组件名称可以通过 setComponent()、setClass() 或 setClassName()方法设置，并通过 getComponent()方法读取。下面分别对上面提到的几个方法进行介绍。

☑ setComponent()方法

setComponent()方法用来为 Intent 设置组件，其语法格式如下：

```
public Intent setComponent(ComponentName component)
```

- component：要设置的组件名称。
- 返回值：Intent 对象。

☑ setClass()方法

setClass()方法用来为 Intent 设置要打开的 Activity，其语法格式如下：

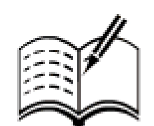
```
public Intent setClass(Context packageContext, Class<?> cls)
```

- packageContext：当前 Activity 的 this 对象。
- cls：要打开的 Activity 的 class 对象。
- 返回值：Intent 对象。

☑ setClassName()方法

setClassName()方法用来为 Intent 设置要打开的 Activity 名称，其语法格式如下：






```
public Intent setClassName(Context packageContext, String className)
```

- packageContext：当前 Activity 的 this 对象。
- className：要打开的 Activity 的类名称。
- 返回值：Intent 对象。

☑ getComponent()方法

getComponent()方法用来获取与 Intent 相关的组件，其语法格式如下：

```
public ComponentName getComponent()
```

返回值为与 Intent 相关的组件名称。

例如，使用 Intent 对象的 setClass()方法设置组件名称，其代码如下：

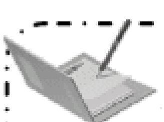
```
Intent intent=new Intent();                                    //创建 Intent 对象
intent.setClass(IntentExamActivity.this, LinkActivity.class); //为 Intent 对象设置组件名称
```

9.2.2　动作

动作很大程度上决定了 Intent 如何构建，特别是数据（data）和附加（extras）信息，就像一个方法名决定了参数和返回值一样，正是由于这个原因，所以应该尽可能明确指定动作，并紧密关联到其他 Intent 字段。也就是说，应该定义组件能够处理的 Intent 对象的整个协议，而不仅仅是单独的定义一个动作。Intent 类定义了一些动作常量，常用的动作常量如表 9.1 所示。

表 9.1　Intent 类的常用动作常量

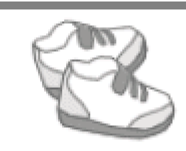
动 作 常 量	作 用 对 象	描　　述
ACTION_CALL	Activity	直接拨打电话
ACTION_DIAL	Activity	打开拨打电话界面
ACTION_VIEW	Activity	查看用户列表
ACTION_EDIT	Activity	编辑用户列表
ACTION_HEADSET_PLUG	Activity	耳机插拔
ACTION_MAIN	Activity	在没有数据输入和输出时，默认的启动 Activity
ACTION_SENDTO	Activity	给某人发送信息
ACTION_SYNC	Activity	使移动设备的数据域服务器数据保持同步
ACTION_BATTERY_LOW	BroadcastReceiver	显示电量低的警告信息
ACTION_SCREEN_ON	BroadcastReceiver	打开屏幕
ACTION_TIMEZONE_CHANGED	BroadcastReceiver	修改时区设置



说明：

Intent 类有很多动作常量，表 9.1 只是给出了常用的一些动作常量，关于 Intent 类的其他动作常量，可以参考 Android 官方帮助文档中的 Intent 类；另外，开发人员还可以定义自己的动作字符串，自定义动作字符串应该包含应用程序包名的前缀，如 com.xiaoke.project.SHOW_ COLOR。





Note

一个 Intent 对象的动作通过 setAction()方法设置，通过 getAction()方法读取。下面分别对 setAction()和 getAction()方法进行介绍。

☑ setAction()方法

setAction()方法用来为 Intent 设置动作，其语法格式如下：

```
public Intent setAction(String action)
```

- ➢ action：要设置的动作名称，通常设置为 Android API 提供的动作常量。
- ➢ 返回值：Intent 对象。

☑ getAction()方法

getAction()方法用来获取 Intent 的动作名称，其语法格式如下：

```
public String getAction()
```

返回值为 String 字符串，表示 Intent 的动作名称。

例如，使用 Intent 对象的 setAction()方法设置 Intent 对象的动作为拨打电话，其代码如下：

```
Intent intent=new Intent();                          //创建 Intent 对象
intent.setAction(Intent.ACTION_CALL);                //设置动作为拨打电话
```

9.2.3 数据

数据（data）是作用于 Intent 上的数据的 URI 和数据的 MIME 类型，不同的动作有不同的数据规格。例如，如果动作字段是 ACTION_EDIT，数据字段应该包含将显示用于编辑的文档的 URI；如果动作是 ACTION_CALL，数据字段应该是一个 tel:URI 和要拨打的号码；如果动作是 ACTION_VIEW，数据字段应该是一个 http:URI。

说明：

当匹配一个 Intent 到一个能够处理数据的组件时，明确其数据类型（它的 MIME 类型）和 URI 很重要。例如，一个组件能够显示图像数据，不应该被调用去播放一个音频文件。

在许多情况下，数据类型能够从 URI 中推测，特别是 content:URIs，它表示位于设备上的数据且被内容提供者（Content Provider）控制，另外，类型也能够显示地设置，使用 setData()方法可以指定数据的 URI，使用 setType()方法可以指定数据的 MIME 类型，使用 setDataAndType()方法可以指定数据的 URI 和 MIME 类型；而通过 getData()方法可以读取取数据的 URI，通过 getType()方法可以读取数据的类型。下面分别对上面提到的几个方法进行介绍。

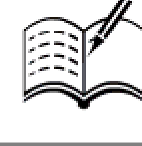
☑ setData()方法

setData()方法用来为 Intent 设置 URI 数据，其语法格式如下：

```
public Intent setData(Uri data)
```

- ➢ data：要设置的数据的 URI。





- 返回值：Intent 对象。

☑ setType()方法

setType()方法用来为 Intent 设置数据的 MIME 类型，其语法格式如下：

```
public Intent setType(String type)
```

- type：要设置的数据的 MIME 类型。
- 返回值：Intent 对象。

☑ setDataAndType()方法

setDataAndType()方法用来为 Intent 设置数据及其 MIME 类型，其语法格式如下：

```
public Intent setDataAndType(Uri data, String type)
```

- data：要设置的数据的 URI。
- type：要设置的数据的 MIME 类型。
- 返回值：Intent 对象。

☑ getData()方法

getData()方法用来获取与 Intent 相关的数据，其语法格式如下：

```
public Uri getData()
```

返回值为 URI 类型，表示获取到的与 Intent 相关数据的 URI。

☑ getType()方法

getType()方法用来获取与 Intent 相关的数据的 MIME 类型，其语法格式如下：

```
public String getType()
```

返回值为 String 字符串，表示获取到的 MIME 类型。

【例 9.1】 在 Eclipse 中创建 Android 项目，主要通过为 Intent 设置动作和数据实现拨打电话和发送短信的功能。

实例位置：光盘\MR\Instance\09\9.1

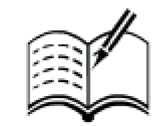
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在其中添加两个 Butotn 组件 btn1 和 btn2，并分别设置它们的文本为“拨打电话”和“发送短信”。其代码如下：

```
<Button
    android:id="@+id/btn1"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:text="拨打电话"
    />
    <Button
    android:id="@+id/btn2"
    android:layout_width="60dp"
```



```
    android:layout_height="40dp"
    android:text="发送短信"
    />
```



Note

（2）打开 MainActivity.java 文件，在 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton1=(Button)findViewById(R.id.btn1);          //获取 btn1 组件
    btnButton1.setOnClickListener(listener);                    //为 btn1 组件设置监听事件
    Button btnButton2=(Button)findViewById(R.id.btn2);          //获取 btn2 组件
    btnButton2.setOnClickListener(listener);                    //为 btn2 组件设置监听事件
}
```

（3）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，通过判断单击的按钮 id，分别为两个 Button 按钮设置拨打电话和发送短信的动作及数据。其代码如下：

```
//创建监听事件对象
private android.view.View.OnClickListener listener=new android.view.View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent=new Intent();                              //创建 Intent 对象
        Button button=(Button)v;                                 //将 View 强制转换为 Button 对象
        switch (button.getId()) {                                //根据 Button 组件的 id 进行判断
        case R.id.btn1:                                          //如果是 btn1 组件
            intent.setAction(Intent.ACTION_CALL);                //设置动作为拨打电话
            intent.setData(Uri.parse("tel:13610780204"));        //设置要拨打的号码
            startActivity(intent);                               //启动 Activity
            break;
        case R.id.btn2:
            intent.setAction(Intent.ACTION_SENDTO);              //设置动作为拨打电话
            intent.setData(Uri.parse("smsto:5554"));             //设置要发送的号码
            intent.putExtra("sms_body", "Welcome to Android!"); //设置要发送的信息内容
            startActivity(intent);                               //启动 Activity
            break;
        }
    }
};
```

（4）打开 AndroidManifest.xml 文件，在其中为当前 Android 程序设置拨打电话和发送短信的权限。其代码如下：

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
```

运行本实例，将显示如图 9.1 所示的运行效果。







图 9.1　主 Activity 界面

单击图 9.1 中的“拨打电话”按钮，进入到拨打电话界面，并开始拨打设置的电话号码，如图 9.2 所示。

单击图 9.1 中的“发送短信”按钮，进入发送信息界面，该界面中自动加载用户设置的号码和要发送的信息，如图 9.3 所示。

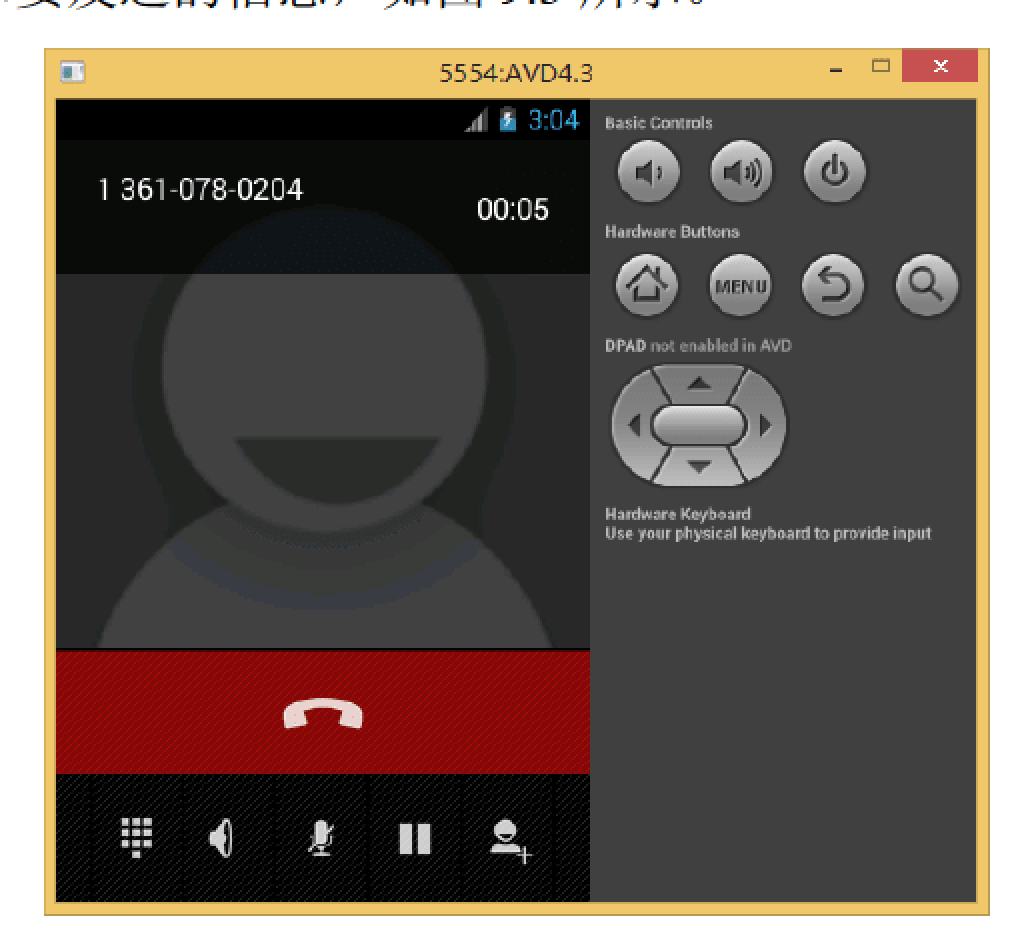




图 9.2　拨打电话

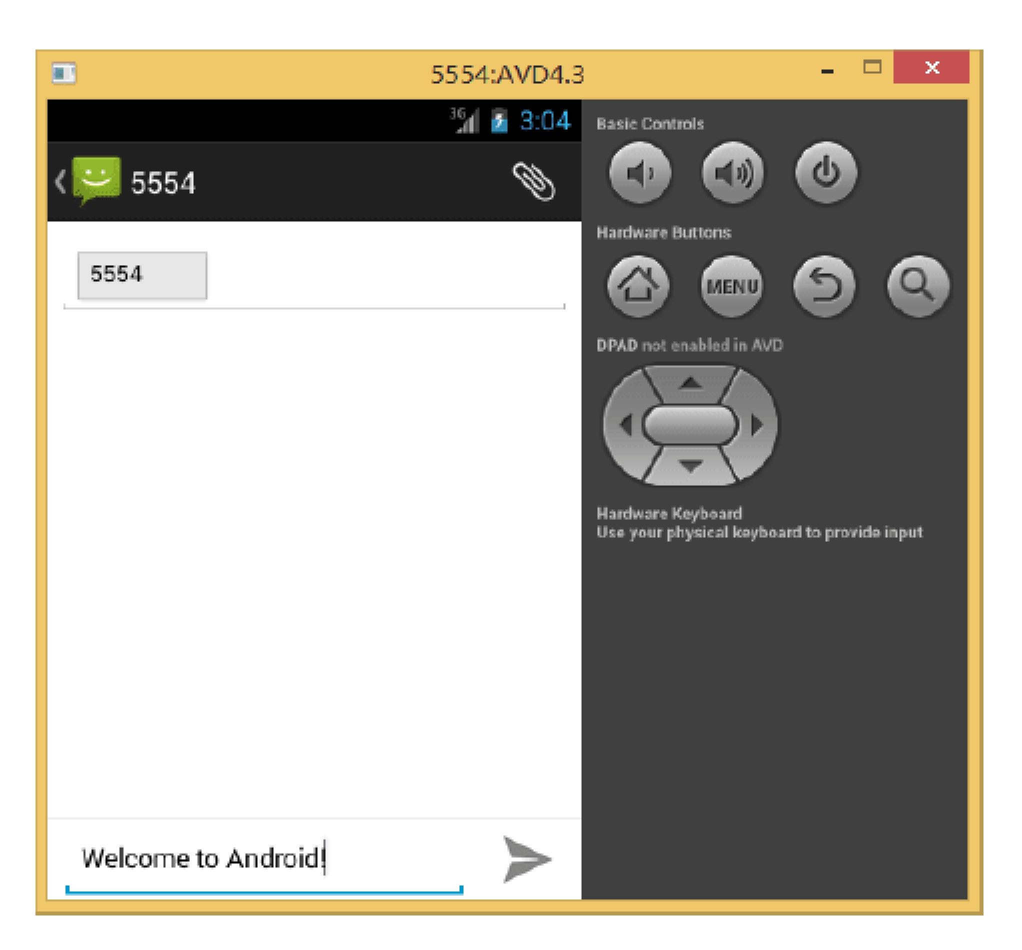


图 9.3　发送信息

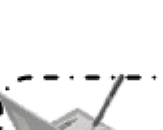
9.2.4　种类

除了组件名称、动作和数据外，Intent 中还可以包含组件类型信息，它用来作为被执行动作的附加信息，开发人员可以在一个 Intent 对象中指定任意数量的种类描述。Intent 类中定义了一些种类常量，常用的种类常量如表 9.2 所示。

表 9.2　Intent 类的常用种类常量

种类常量	描　述
CATEGORY_DEFAULT	默认的种类常量
CATEGORY_ALTERNATIVE	表示当前的 Intent 是一系列可选动作中的一个，这些动作可以在同一块数据上执行
CATEGORY_BROWSABLE	表示浏览器在特定条件下可以打开 Activity
CATEGORY_GADGET	表示当前 Activity 可以被嵌入到充当配件宿主的其他 Activity 中
CATEGORY_HOME	表示 Activity 将显示桌面，即：用户开机后看到的第一个屏幕，或者按 HOME 键时看到的屏幕
CATEGORY_LAUNCHER	表示 Intent 的接受者应该在 Launcher 中作为顶级应用出现
CATEGORY_PREFERENCE	表示目标 Activity 是一个选择面板





Note

说明：

Intent 类有很多种类常量，表 9.2 只是给出了常用的一些种类常量，关于 Intent 类的其他种类常量，可以参考 Android 官方帮助文档中的 Intent 类。

在 Android 程序开发中，可以使用 addCategory()方法添加一个种类到 Intent 对象中，使用 removeCategory()方法删除一个之前添加的种类，使用 getCategories()方法获取 Intent 对象中的所有种类。下面分别对上面提到的几个方法进行介绍。

☑ addCategory()方法

addCategory()方法用来为 Intent 添加种类信息，其语法格式如下：

```
public Intent addCategory(String category)
```

- category：要添加的种类信息，通常用 Android API 中提供的种类常量表示。
- 返回值：Intent 对象。

☑ removeCategory()方法

removeCategory()方法用来从 Intent 中删除指定的种类信息，其语法格式如下：

```
public void removeCategory(String category)
```

category 表示要删除的种类信息。

☑ getCategories()方法

getCategories()方法用来获取所有与 Intent 相关的种类信息，其语法格式如下：

```
public Set<String> getCategories()
```

返回值为字符串类型的泛型数组，表示所有与 Intent 相关的种类信息。

例如，创建 Intent 对象，并为其设置种类常量，其代码如下：

```
Intent intent=new Intent();                                    //创建 Intent 对象
intent.addCategory(Intent.CATEGORY_LAUNCHER);                  //为 Intent 对象添加种类信息
```

9.2.5 附加信息

额外的键值对信息应该传递到组件处理 Intent，就像动作关联的特定种类的数据 URIs，也关联到某些特定的附加信息。例如，一个 ACTION_TIMEZONE_CHANGE 动作有一个“time-zone”的附加信息，标识新的时区；ACTION_HEADSET_PLUG 动作有一个“state”附加信息，标识头部现在是否塞满或未塞满，有一个“name”附加信息，标识头部的类型。

Intent 对象中有一系列的 putXX()方法用于插入各种附加数据，一系列的 getXX()方法用于读取数据，这些方法与 Bundle 对象的方法类似，实际上，附加信息可以作为一个 Bundle 对象使用 putExtras()和 getExtras()方法安装和读取。下面分别对 putExtras()和 getExtras()方法进行介绍。

☑ putExtras()方法

putExtras()方法用来为 Intent 添加附加信息，该方法有多种重载形式，其常用的一种重载形



式如下：

```
public Intent putExtra(String name, String value)
```

- name：附加信息的名称。
- value：附加信息的值。
- 返回值：Intent 对象。

☑ getExtras()方法

getExtras()方法用来获取 Intent 中的附加信息，其语法格式如下：

```
public Bundle getExtras()
```

返回值为 Bundle 对象，用来存储获取到的 Intent 附加信息。



技巧：

putExtras()和 getExtras()方法通常用来在多个 Activity 之间传值。

【例 9.2】 在 Eclipse 中创建 Android 项目，主要使用 putExtras()和 getExtras()方法实现为 Intent 添加附加信息和读取附加信息的功能。

实例位置：光盘\MR\Instance\09\9.2

程序的开发步骤如下：

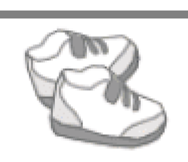
（1）创建一个 Activity，命名为 AcceptdataActivity，并在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btn，并设置其文本为“跳转”。其代码如下：

```
<Button
    android:id="@+id/btn"
    android:layout_width="60dp"
    android:layout_height="40dp"
    android:text="跳转"
    />
```

（3）在 res\layout 目录下创建一个 link.xml 文件，用来作为 AcceptdataActivity 的布局文件，在该布局文件中添加一个 TextView 组件。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:id="@+id/txt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="链接页面"
```



```
        />
</LinearLayout>
```

（4）打开 MainActivity.java 文件，定义一个 int 类型常量，用来作为请求标识。其代码如下：

```
private final static int REQUEST_CODE=1;                    //声明请求标识
```

（5）在 MainActivity.java 文件的 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button btnButton=(Button) findViewById(R.id.btn);        //获取 Button 按钮
    btnButton.setOnClickListener(listener);                  //为 Button 按钮设置监听事件
}
```

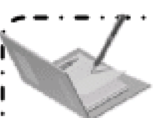
（6）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，首先创建一个 Intent 对象，并设置要打开的 Activity，然后使用 Intent 对象的 putExtra()方法设置附加信息，最后使用 startActivityForResult()方法启动 Activity。其主要代码如下：

```
private OnClickListener listener=new OnClickListener() {            //创建监听对象
    @Override
    public void onClick(View v) {
        Intent intent=new Intent();                                 //创建 Intent 对象
        //设置要访问的 Activity
        intent.setClass(MainActivity.this, AcceptdataActivity.class);
        intent.putExtra("str", "第一个 Activity 传过来的值");         //设置附加信息
        startActivityForResult(intent, REQUEST_CODE);               //启动 Activity
    }
};
```

（7）打开 AcceptdataActivity.java 文件，在 onCreate()方法中，使用 Intent 对象的 getExtras()方法获取附加信息，并显示到 TextView 组件中。onCreate()方法代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.link);
    Intent intent=getIntent();                              //创建 Intent 对象
    Bundle bundle=intent.getExtras();                       //获取附加信息，并用 Bundle 接收
    String str=bundle.getString("str");                     //获取传递的字符串值
    txt=(TextView)findViewById(R.id.txt);                   //获取 TextView 组件
    txt.setText(str);                                       //设置文本
}
```





说明：

AcceptdataActivity.java 文件使用 link.xml 作为布局文件。

运行本实例，将显示如图 9.4 所示的运行效果。单击“跳转”按钮，进入第二个 Activity，该 Activity 窗口中显示第一个 Activity 中设置的附加信息，如图 9.5 所示。



图 9.4　主 Activity 初始页面



图 9.5　获取到的附加信息

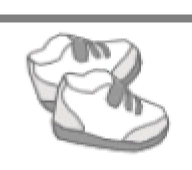
9.2.6　标志

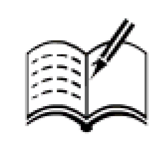
标志主要用来指示 Android 程序如何去启动一个活动（如活动应该属于那个任务）和启动之后如何对待它（如它是否属于最近的活动列表），所有的标志都定义在 Intent 类中。常用的标志常量如表 9.3 所示。

表 9.3　Intent 类的常用标志常量

标 志 常 量	描　述
FLAG_GRANT_READ_URI_PERMISSION	对 Intent 数据具有读取权限
FLAG_GRANT_WRITE_URI_PERMISSION	对 Intent 数据具有写入权限
FLAG_ACTIVITY_CLEAR_TOP	如果在当前 Task 中有要启动的 Activity，那么把该 Activity 之前的所有 Activity 都关掉，并把该 Activity 置前以避免创建 Activity 的实例
FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET	如果设置，将在 Task 的 Activity Stack 中设置一个还原点，当 Task 恢复时，需要清理 Activity
FLAG_ACTIVITY_EXCLUDE_FROM_RECENTS	如果设置，新的 Activity 不会在最近启动的 Activity 的列表中保存
FLAG_ACTIVITY_FORWARD_RESULT	如果设置，并且这个 Intent 用于从一个存在的 Activity 启动一个新的 Activity，那么，这个作为答复目标的 Activity 将会传到这个新的 Activity 中。这种方式下，新的 Activity 可以调用 setResult(int)，并且这个结果值将发送给那个作为答复目标的 Activity
FLAG_ACTIVITY_LAUNCHED_FROM_HISTORY	这个标志一般不由应用程序代码设置，如果这个 Activity 是从历史记录里启动的（按 HOME 键），那么，系统会自动设定
FLAG_ACTIVITY_MULTIPLE_TASK	与 FLAG_ACTIVITY_NEW_TASK 结合使用，使用时，新的 Task 总是会启动来处理 Intent，而不管是否已经有一个 Task 可以处理相同的事情
FLAG_ACTIVITY_NEW_TASK	系统会检查当前所有已创建的 Task 中是否有需要启动的 Activity 的 Task，如果有，则在该 Task 上创建 Activity；如果没有，则新建具有该 Activity 属性的 Task，并在该新建的 Task 上创建 Activity

Note





Note



标志常量	描　述
FLAG_ACTIVITY_NO_HISTORY	新的 Activity 将不再历史 Stack 中保留，用户一旦离开他，这个 Activity 自动关闭
FLAG_ACTIVITY_NO_USER_ACTION	如果设置，作为新启动的 Activity 进入前台时，这个标志将在 Activity 暂停之前阻止从最前方的 Activity 回调的 onUserLeaveHint()方法



说明：

Intent 类有很多标志常量，表 9.3 只是给出了常用的一些标志常量，关于 Intent 类的其他标志常量，可以参考 Android 官方帮助文档中的 Intent 类。

注意：

由于默认的系统不包含图形 Task 管理功能，因此，尽量不要使用 FLAG_ACTIVITY_MULTIPLE_TASK 标志，除非能够提供给用户一种方式——可以返回到已经启动的 Task。

在 Android 程序开发中，可以使用 setFlags()和 addFlags()方法添加一个标志到 Intent 对象中，使用 getFlags()方法获取 Intent 对象中的所有标志。下面分别对上面提到的几个方法进行介绍。

☑ setFlags()方法

setFlags()方法用来为 Intent 设置标志，其语法格式如下：

```
public Intent setFlags(int flags)
```

- flags：要设置的标志，通常用 Android API 中提供的标志常量表示。
- 返回值：Intent 对象。

☑ addFlags()方法

addFlags()方法用来为 Intent 添加标志，其语法格式如下：

```
public Intent addFlags(int flags)
```

- flags：要添加的标志，通常用 Android API 中提供的标志常量表示。
- 返回值：Intent 对象。

☑ getFlags()方法

getFlags()方法用来获取 Intent 的标志，其语法格式如下：

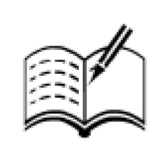
```
public int getFlags()
```

返回值为 int 类型数据，表示获取到的标志。

例如，创建 Intent 对象，并为其设置标志信息，其代码如下：

```
Intent intent=new Intent();                          //创建 Intent 对象
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);     //为 Intent 对象添加标志信息
```





9.3 解析 Intent 对象

Intent 可以分为两组，分别为显式 Intent 和隐式 Intent。

☑ 显式 Intent：通过名字指定目标组件。因为开发者通常不知道其他应用程序的组件名字，显式 Intent 通常用于应用程序内部消息，如一个活动启动从属的服务或启动一个姐妹活动。

☑ 隐式 Intent：并不指定目标的名字（组件名字字段是空的）。隐式 Intent 经常用于激活其他应用程序中的组件。

当在 Android 程序中使用显式 Intent 时，Intent 对象中只用组件名字内容就可以决定哪个组件应该获得这个 Intent，而不用其他内容。而使用隐式 Intent 时，由于默认指定目标，Android 程序必须查找一个最适合的组件（一些组件）去处理 Intent——一个活动或服务去执行请求动作，或一组广播接收者去响应广播声明，该过程是通过比较 Intent 对象的内容和 Intent 过滤器（intent filters）来完成的。Intent 过滤器关联到潜在的接收 Intent 的组件，过滤器声明组件的能力和界定它能处理的 Intents，它们打开组件接收声明的 Intent 类型的隐式 Intents。如果一个组件没有任何 Intent 过滤器，它仅能接收显式的 Intents，而声明了 Intent 过滤器的组件可以接收显式和隐式的 Intents。本节将对如何解析 Intent 对象进行详细讲解。



说明：

只有当一个 Intent 对象的下面 3 个方面都符合一个 Intent 过滤器：动作、数据（包括 URI 和数据类型）和种类，才被考虑是否接收 Intent，而附加信息和标志在解析哪个组件接收 Intent 时不起作用。

9.3.1 Intent 过滤器

活动、服务、广播接收者为了告知系统能够处理哪些隐式 Intent，它们可以有一个或多个 Intent 过滤器，Intent 过滤器用<intent-filter>元素表示，每个过滤器描述组件的一种能力，即能够接收的一组 Intent。实际上，它筛选掉的 Intents 也仅仅是不想要的隐式 Intents。一个显式 Intent 不管包含哪些信息，总是能够传递到它的目标组件；但是一个隐式 Intent，仅当它能够通过组件的过滤器之一才能够传递给它。

一个组件能够做的每一个任务都有独立的过滤器。例如，记事本中的 NoteEditer 活动有两个过滤器，一个是启动一个指定的记录，用户可以查看和编辑；另一个是启动一个新的、空的记录，用户能够填充并保存。

一个 Intent 过滤器是一个 IntentFilter 类的实例，因为 Android 程序在启动一个组件之前必须知道它的能力，但是 Intent 过滤器通常不在 Java 代码中设置，而是在应用程序的主配置文件（AndroidManifest.xml）中以<intent-filter>元素设置。





说明：

广播接收者的过滤器通过调用 Context.registerReceiver()方法动态地注册，它直接创建一个 IntentFilter 对象。

Note

一个过滤器有对应于 Intent 对象的动作、数据、种类的字段，过滤器要检测隐式 Intent 的所有这 3 个字段，其中任何一个失败，Android 程序都不会传递 Intent 给组件，然而，因为一个组件可以有多个 Intent 过滤器，一个 Intent 通不过组件的过滤器检测，其他过滤器可能通过检测。下面分别对如何使用 Intent 过滤器进行动作、数据和种类的检测进行讲解。

1. 动作检测

在 AndroidManifest.xml 主配置文件中，使用<intent-filter>元素的<action>子元素列出所有动作信息。

例如，使用 intent-filter 过滤器的<action>子元素设置主窗口启动、拨打电话和发送信息等 3 个动作，其代码如下：

```
<intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <action android:name="android.intent.action.CALL"/>
        <action android:name="android.intent.action.SENDTO"/>
. . .
</intent-filter>
```

如上面示例所示，虽然一个 Intent 对象仅是单个动作，但是一个 intent-filter 过滤器可以列出不止一个，另外，这个动作列表不能够为空，一个 intent-filter 过滤器必须至少包含一个<action>子元素，否则它将阻塞所有的 Intents。

要通过动作检测，Intent 对象中指定的动作必须匹配 intent-filter 过滤器的动作列表中的一个，而如果 Intent 对象或 intent-filter 过滤器没有指定一个动作，结果将如下。

- ☑ 如果 Intent 对象没有指定动作，将自动通过检查<intent-filter>过滤器是否指定了动作。
- ☑ 如果 intent-filter 过滤器没有指定动作，没有一个 Intent 相匹配，所有的 Intent 将检测失败，即没有 Intent 能够通过过滤器。

2. 数据检测

在 AndroidManifest.xml 主配置文件中，使用<intent-filter>元素的<data>子元素列出所有数据信息。

例如，使用 intent-filter 过滤器的<data>子元素设置两个数据信息，其代码如下：

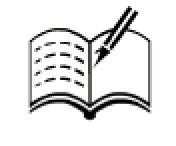
```
<intent-filter>
      <data android:mimeType="video/mpeg" android:scheme="http"/>
      <data android:mimeType="audio/mpeg" android:scheme="http"/>
. . .
</intent-filter>
```

如上面示例所示，每个<data>子元素都需要指定一个数据类型（MIME 类型）和 URI。<data>



子元素中有 scheme、host、port 和 path 4 个属性，分别对应于 URI 的每个部分。例如，下面的 URI：

```
content://com.xiaoke.project:80/folder/subfolder/etc
```



如果在<data>中指定上面的 URI，则 scheme 对应 content，host 对应 com.xiaoke.project，port 对应 80，path 对应 folder/subfolder/etc。host 和 port 一起构成 URI 的凭据（authority），如果 host 没有指定，port 也被忽略。这 4 个属性都是可选的，但它们之间并不都是完全独立的，例如，要使 authority 有意义，scheme 必须也要指定；而要使 path 有意义，scheme 和 authority 也都必须要指定。



说明：

当比较 Intent 对象和 intent-filter 过滤器的 URI 时，仅仅比较 intent-filter 过滤器中出现的 URI 属性。例如，如果一个 intent-filter 过滤器仅指定了 scheme，所有包含该 scheme 的 URIs 都将匹配 intent-filter 过滤器；如果一个 intent-filter 过滤器指定了 scheme 和 authority，但没有指定 path，所有匹配 scheme 和 authority 的 URIs 都可以通过检测，而不管它们的 path 是否匹配；如果 4 个属性都指定了，就需要都匹配才能算是匹配，然而，intent-filter 过滤器中的 path 可以包含通配符来要求匹配 path 中的一部分。

<data>子元素中的 type 属性用来指定数据的 MIME 类型。Intent 对象和 intent-filter 过滤器都可以用“*”通配符匹配类型字段，如“text/*”、“audio/*”表示任何子类型。

在 Android 程序中进行数据检测时，既要检测 URI，也要检测数据类型。检测规则如下：

- ☑ 一个 Intent 对象既不包含 URI，也不包含数据类型：仅当 intent-filter 过滤器也不指定任何 URIs 和数据类型时，才不能通过检测；否则都能通过。
- ☑ 一个 Intent 对象包含 URI，但不包含数据类型：仅当 intent-filter 过滤器也不指定数据类型，同时它们的 URI 匹配，才能通过检测。例如，mailto:和 tel:都不指定实际数据。
- ☑ 一个 Intent 对象包含数据类型，但不包含 URI：仅当 intent-filter 过滤器也只包含数据类型且与 Intent 相同时，才通过检测。
- ☑ 一个 Intent 对象既包含 URI，也包含数据类型（或数据类型能够从 URI 推断出）：数据类型部分，只有与 intent-filter 过滤器中之一匹配才算通过；URI 部分，它的 URI 要出现在 intent-filter 过滤器中，或者它有 content:或 file: URI，又或者 intent-filter 过滤器没有指定 URI。

3. 种类检测

在 AndroidManifest.xml 主配置文件中，使用<intent-filter>元素的<category>子元素列出所有种类信息。例如，使用 intent-filter 过滤器的<category>子元素设置 CATEGORY_DEFAULT 和 CATEGORY_LAUNCHER 两个种类信息，其代码如下：

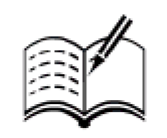
```
<intent-filter>
    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.LAUNCHER"/>
...
</intent-filter>
```





说明：

上面示例中所示的"android.intent.category.DEFAULT"字符串对应于表 9.2 中的 CATEGORY_DEFAULT 常量，"android.intent.category.LAUNCHER"字符串对应于表 9.2 中的 CATEGORY_LAUNCHER 常量。



Note

对于一个 Intent 对象，如果要通过种类检测，Intent 对象中的每个种类必须匹配过滤器中的一个，即过滤器能够列出额外的种类，但是 Intent 对象中的种类都必须能够在过滤器中找到，即使只有一个种类在过滤器列表中没有，就算种类检测失败。因此，原则上如果一个 Intent 对象中没有种类（即种类字段为空），应该总是通过种类测试，而不管 intent-filter 过滤器中有什么种类，但是有个例外，Android 程序对待所有传递给 Context.startActivity()的隐式 Intent，都至少包含 android.intent.category.DEFAULT（对应 CATEGORY_DEFAULT 常量），因此，Activity 如果要接收隐式 Intent，就必须在 intent-filter 过滤器中包含 android.intent.category.DEFAULT。

9.3.2 通用情况

在 Android 程序中对 Intent 对象进行解析时，主要有两种通用情况，分别如下。

☑ 在过滤器中只指定数据类型

在讲解数据检测规则时，有一条规则是：一个 Intent 对象既包含 URI，也包含数据类型（或数据类型能够从 URI 推断出）时，数据类型部分，只有与 intent-filter 过滤器中之一匹配才算通过；URI 部分，它的 URI 要出现在 intent-filter 过滤器中，或者它有 content:或 file:URI，又或者 intent-filter 过滤器没有指定 URI。这条规则表明组件能够从内容提供者或文件获取本地数据，因此，它们的过滤器仅列出数据类型，而不必明确指出 content:和 file:scheme 的名字。

例如，在 intent-filter 过滤器中使用<data>子元素只指定数据类型，其代码如下：

```
<data android:mimeType="image/*" />
```

上面的代码告诉 Android 程序：这个组件能够从内容提供者获取 image 数据并显示它，因为大部分可用数据由内容提供者（Content Provider）分发，所以过滤器指定一个数据类型，但没有指定 URI 或许最通用。

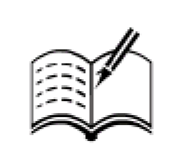
☑ 在过滤器中指定一个 scheme 和一个数据类型

例如，在 intent-filter 过滤器中使用<data>子元素指定一个 scheme 和一个数据类型，其代码如下：

```
<data android:scheme="http" android:type="video/*" />
```

上面的代码告诉 Android 程序：这个组件能够从网络获取视频数据并显示它。





9.3.3 使用 Intent 匹配

Intent 对照着 intent-filter 过滤器匹配，不仅去发现一个目标组件去激活，而且去发现设备上的组件的其他信息。例如，Android 程序填充应用程序启动列表，最高层屏幕显示用户能够启动的应用程序，该功能的实现过程是：首先查找所有包含 android.intent.action.MAIN 动作和 android.intent.category.LAUNCHER 种类的过滤器，然后在启动列表中显示这些应用程序的图标和标签。类似的，Android 程序可以通过查找含有 android.intent.category.HOME 过滤器的活动来发掘主菜单。

9.4 使用 Intent 传递数据

Intent 对象是一个被动的数据结构保存一个将要执行的操作的抽象描述，或在广播的情况下，通常是某事已经发生并正在执行，它通常用来激活 Activity、Service 和 BroadcastReceiver，并在它们之间传递数据。本节将对如何使用 Intent 在 Activity 间传递数据进行详细讲解。

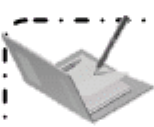
9.4.1 无参数 Activity 跳转

无参数 Activity 跳转功能的实现使用 startActivity()方法实现，该方法的语法中有一个 Intent 对象作为参数，用来表示要执行的 Intent，该 Intent 必须已经设置了要跳转到的 Activity 名称。startActivity()方法语法格式如下：

```
public void startActivity(Intent intent)
```

例如，使用 startActivity()方法实现无参数 Activity 跳转功能，其代码如下：

```
Intent it = new Intent(Activity.Main.this, Activity2.class);        //创建 Intent 对象
startActivity(it);                                                    //跳转 Activity
```

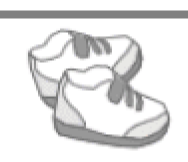


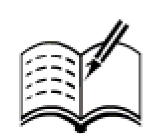
说明：

上面代码中的 Activity.Main 是当前的 Activity 名称，Activity2 是要跳转到的 Activity 名称。

9.4.2 向下一个 Activity 传递数据

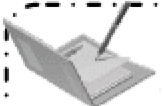
向下一个 Activity 传递数据主要有两种情况，分别是将数据传递到打开的 Activity 中和得到新打开 Activity 关闭后返回的数据。下面分别对这两种传递数据的方式进行讲解。




Note

1．将数据传递到打开的 Activity 中

将数据传递到要打开的 Activity 中时，首先需要使用 putExtra()方法设置要传递的数据，然后使用相应的 getXX Extra()方法获取传递的数据。由于在 Activity 间传递数据时，可以传递多种类型的数据，如 String 类型、Int 类型和 Bool 类型等，而根据传递数据类型的不同，获取传递的数据的方法也不同，例如，如果传递的是 String 类型数据，则使用 getStringExtra()方法获取传递的数据；如果传递的是 Int 类型数据，则使用 getIntExtra()方法获取传递的数据；如果传递的是 Bool 类型数据，则使用 getBooleanExtra()方法获取传递的数据。


说明：

如果不确定传递的数据类型，可以直接使用 getExtras()方法获取传递的数据，但是使用该方法获取到的数据是存储在 Bundle 对象中的。

【例 9.3】 在 Eclipse 中创建 Android 项目，主要实现将字符串数据传递到打开的 Activity 中的功能。

实例位置：光盘\MR\Instance\09\9.3

程序的开发步骤如下：

（1）新建一个 Activity，命名为 AcceptActivity，并在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btn，并设置其文本为“传递数据”。其代码如下：

```
<Button
      android:id="@+id/btn"
      android:layout_width="120dp"
      android:layout_height="40dp"
      android:text="传递数据"
      />
```

（3）在 res\layout 目录下创建一个 accept.xml 文件，用来作为 AcceptActivity 的布局文件，在该布局文件中添加一个 TextView 组件。其主要代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="vertical" >
     <TextView
          android:id="@+id/txt"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/hello" />
</LinearLayout>
```

（4）打开 MainActivity.java 文件，在 onCreate()方法中，获取布局文件中的 Button 按钮，并





为其设置单击监听事件。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button btnButton=(Button) findViewById(R.id.btn);                //获取 Button 按钮
        btnButton.setOnClickListener(listener);                          //为 Button 按钮设置监听事件
}
```

（5）上面的代码中用到了 listener 对象，该对象为 OnClickListener 类型，因此在 Activity 中创建该对象，并重写其 onClick()方法。在该方法中，首先创建一个 Intent 对象，并设置要打开的 Activity，然后使用 Intent 对象的 putExtra()方法设置要传递的数据，最后使用 startActivity()方法启动 Activity。其主要代码如下：

```
private OnClickListener listener=new OnClickListener() {              //创建监听对象
        @Override
        public void onClick(View v) {
                //TODO Auto-generated method stub
                Intent intent=new Intent();                              //创建 Intent 对象
                intent.setClass(MainActivity.this, AcceptActivity.class);  //设置要访问的 Activity
                intent.putExtra("str", "将数据传递到打开的 Activity 中");   //设置要传递的值
                startActivity(intent);                                   //启动 Activity
        }
};
```

（6）打开 AcceptActivity.java 文件，覆写 onCreate()方法。在该方法中，首先设置其使用的布局文件，然后使用 Intent 对象的 getStringExtra()方法获取传递的字符串数据，最后将获取到的字符串数据显示在 TextView 组件中。onCreate()方法的代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        //TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.accept);                                 //设置布局文件
        Intent intent=getIntent();                                       //创建 Intent 对象
        String str=intent.getStringExtra("str");                         //获取传递的数据
        TextView txt=(TextView)findViewById(R.id.txt);                   //获取 TextView 组件
        txt.setText(str);                                                //将获取的数据显示在 TextView 中
}
```

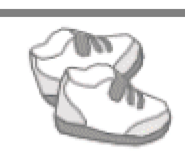
运行本实例，将显示如图 9.6 所示的运行效果。单击“传递数据”按钮，进入第二个 Activity，在该 Activity 的 TextView 组件中显示第一个 Activity 传过来的字符串数据，如图 9.7 所示。

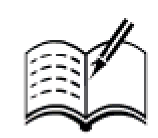


图 9.6　主 Activity 页面



图 9.7　接收到的字符串数据





2．得到新打开 Activity 关闭后返回的数据

如果要在 Activity 中得到新打开 Activity 关闭后返回的数据，首先需要使用系统提供的 startActivityForResult(Intent intent, int requestCode)方法打开新的 Activity，然后在新打开的 Activity 关闭前，使用 setResult(int resultCode, Intent data)方法向前面的 Activity 返回数据，最后为了得到返回的数据，需要在前面的 Activity 中重写 onActivityResult(int requestCode, int resultCode, Intent data)方法实现。下面分别对 startActivityForResult(Intent intent, int requestCode)、setResult(int resultCode, Intent data)和 onActivityResult(int requestCode, int resultCode, Intent data)方法进行详细介绍。

☑ startActivityForResult(Intent intent, int requestCode)方法

startActivityForResult(Intent intent, int requestCode)方法用来以带有返回值的方式启动新的 Activity，其语法格式如下：

```
public void startActivityForResult(Intent intent, int requestCode)
```

- intent：要启动的 Intent 对象。
- requestCode：请求码，该值是根据业务需要由自已设定，用于标识请求来源。

☑ setResult(int resultCode, Intent data)方法

setResult(int resultCode, Intent data)方法用来为要返回到的 Avtivity 设置结果码，其语法格式如下：

```
public final void setResult(int resultCode, Intent data)
```

- resultCode：结果码，该值是根据业务需要由自已设定，该值通常采用 RESULT_CANCELED 或者 RESULT_OK 表示。
- data：要返回到的 Activity 所在的 Intent 对象。

☑ onActivityResult(int requestCode, int resultCode, Intent data)方法

onActivityResult(int requestCode, int resultCode, Intent data)方法用来获取请求码和结果码以获取新 Activity 中返回的数据，其语法格式如下：

```
public void onActivityResult(int requestCode, int resultCode, Intent data)
```

- requestCode：请求码，即调用 startActivityForResult()方法传递过去的值。
- resultCode：结果码，用于标识返回数据来自哪个新 Activity。
- data：Intent 对象，用来取出新 Activity 返回的数据。

【例 9.4】 在 Eclipse 中创建 Android 项目，主要实现得到新打开 Activity 关闭后返回数据的功能。

实例位置：光盘\MR\Instance\09\9.4

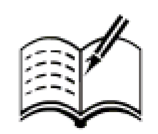
程序的开发步骤如下：

（1）新建一个 Activity，命名为 NewActivity，并在 AndroidManifest.xml 文件中进行配置。

（2）在布局文件 main.xml 中添加一个 Butotn 组件 btn，并设置其文本为“打开 NewActivity”，



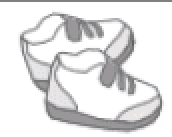
该组件用来打开新的 Activity；添加一个 TextView 组件和一个 EditText 组件，其中，TextView 组件用来作为一个标识，而 EditText 组件用来显示得到的返回数据。main.xml 布局文件的代码如下：



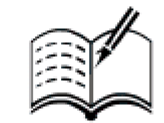

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="打开 NewActivity"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="接收的返回值："
        />
    <EditText
        android:id="@+id/txtData"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

（3）在 res\layout 目录下创建一个 newlayout.xml 文件，用来作为 NewActivity 的布局文件。在该布局文件中添加一个 TextView 组件、一个 EditText 组件和一个 Butotn 组件，其中，TextView 组件用来作为一个标识；EditText 组件用来输入要返回的数据；Butotn 组件用来执行返回操作，并将返回数据传递到第一个 Activity 中。newlayout.xml 布局文件的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入要返回的值：" />
    <EditText
        android:id="@+id/txtBack"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
    <Button
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:text="返回数据"
            />
</LinearLayout>
```



Note

（4）打开 MainActivity.java 文件，定义一个 int 类型常量，用来作为请求标识。其代码如下：

```
protected static final int REQUEST_CODE = 0;                    //声明请求标识
```

（5）在 MainActivity.java 文件的 onCreate()方法中，获取布局文件中的 Button 按钮，并为其设置单击监听事件。在监听事件中，首先创建一个 Intent 对象，并设置要打开的 Activity，然后使用 startActivityForResult()方法启动 Activity。其代码如下：

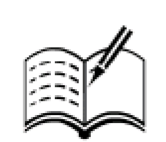
```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button button=(Button) findViewById(R.id.btnOpen);          //获取布局文件中的 Button 组件
    button.setOnClickListener(new OnClickListener() {           //为 Button 组件设置监听事件
        @Override
        public void onClick(View v) {
            //创建 Intent 对象
            Intent intent=new Intent(MainActivity.this, NewActivity.class);
            //以带有返回结果的方式启动 Activity
            startActivityForResult(intent, REQUEST_CODE);
        }
    });
}
```

（6）步骤（5）中启动了 NewActivity，打开该 Activity，在其 onCreate()方法中，首先为该 Activity 设置布局文件，然后获取 EditText 和 Button 组件，并为 Button 组件设置单击监听事件。在监听事件中，首先创建一个 Intent 对象，并使用 Intent 对象的 putExtra()方法将 EditText 组件中的输入设置为要返回的值，然后使用 setResult()方法设置返回标识，最后使用 finish()方法关闭当前 Activity。NewActivity 的 onCreate()方法代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    //TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.newlayout);                          //设置布局文件
    final EditText txt=(EditText) findViewById(R.id.txtBack);    //获取 EditText 组件
    Button btn=(Button) findViewById(R.id.btnBack);              //获取布局文件中的 Button 组件
    btn.setOnClickListener(new OnClickListener() {               //为 Button 组件设置监听事件
        @Override
        public void onClick(View v) {
            //TODO Auto-generated method stub
            Intent intent=new Intent();                          //创建 Intent 对象
            intent.putExtra("back", txt.getText().toString());   //设置返回值
            setResult(Activity.RESULT_OK, intent);               //设置返回标识
```



```
                finish();                                    //关闭当前 Activity
            }
        });
}
```



（7）返回到 MainActivity.java 文件，该文件中重写 onActivityResult()方法，并在该方法中实现获取 Activity 返回值的功能。其代码如下：

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    //TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==REQUEST_CODE){                              //判断请求标识
        if(resultCode==Activity.RESULT_OK){                     //判断结果标识
            Bundle bundle=data.getExtras();                     //获取返回值，并用 Bundle 接收
            String str=bundle.getString("back");                //获取 Bundle 中的返回值
            EditText txt=(EditText) findViewById(R.id.txtData); //获取 EditText 组件
            txt.setText(str);                                   //将返回值显示到 EditText 中
        }
    }
}
```

运行本实例，将显示如图 9.8 所示的运行效果。单击“打开 NewActivity”按钮，进入第二个 Activity，如图 9.9 所示。



图 9.8　主 Activity 初始页面

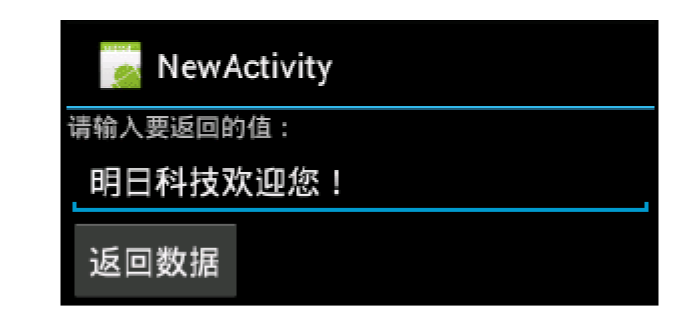


图 9.9　第二个 Activity

在图 9.9 所示的第二个 Activity 中输入要返回的值，单击“返回数据”按钮，返回到第一个 Activity，并在第一个 Activity 的 EditText 组件中显示接收的返回值，如图 9.10 所示。



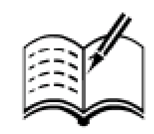
图 9.10　在第一个 Activity 中显示接收的返回值

9.5　综　合　应　用

9.5.1　使用 Intent 实现直接发送短信

【例 9.5】 本实例将使用 Intent 实现发送短信的功能，运行程序，输入电话号码和信息内容，





Note

效果如图 9.11 所示。单击“发送短信”按钮，跳转到如图 9.12 所示的界面，该界面中已经填写好了发送短信的接收者号码及短信内容。

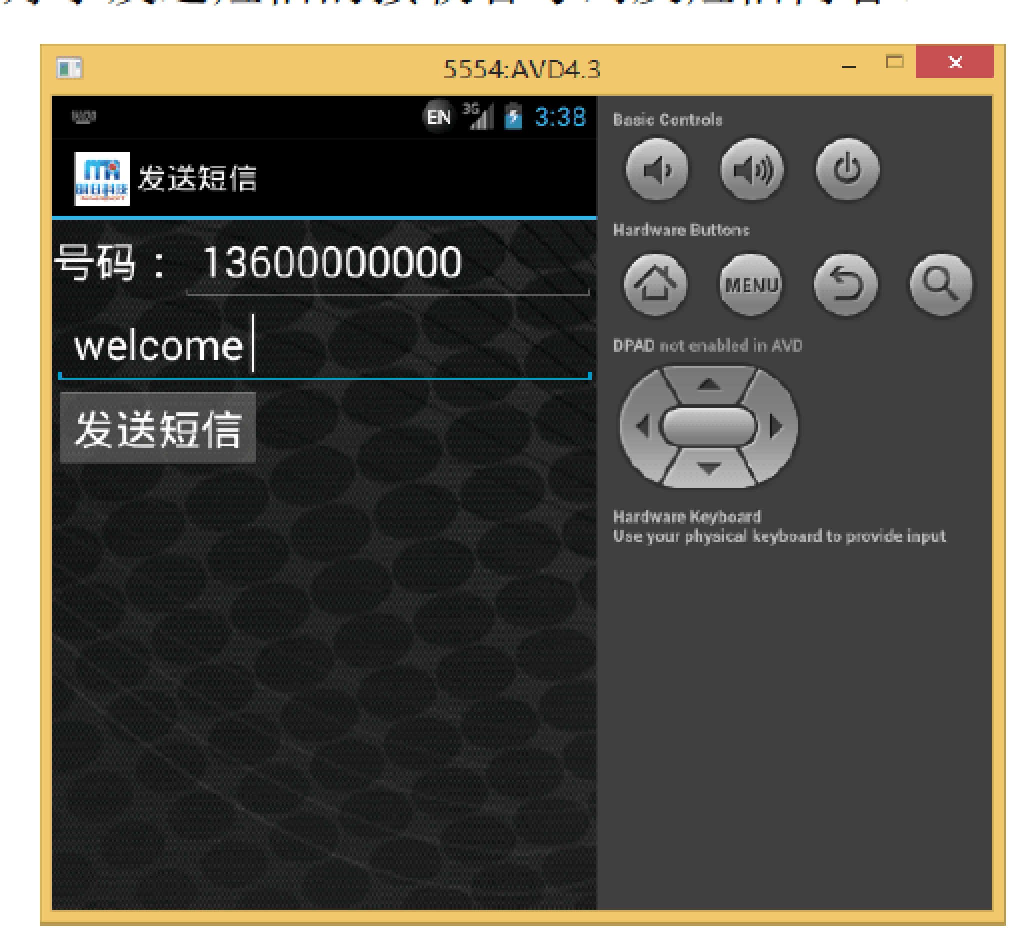


图 9.11　应用程序界面

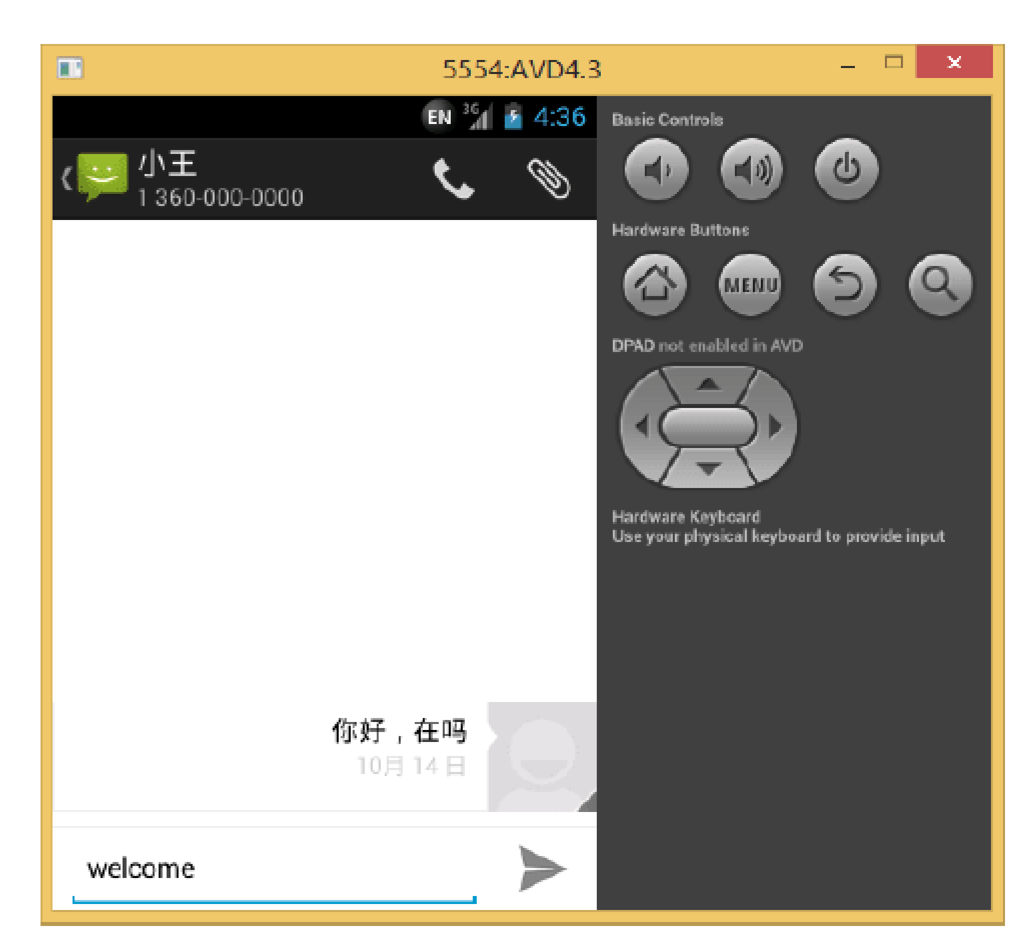


图 9.12　发送短信界面

实例位置：光盘\MR\Instance\09\9.5

程序的开发步骤如下：

（1）在 res\layout 文件夹中打开布局文件 main.xml，增加文本框、按钮等控件，并修改其默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/number"
            android:textColor="@android:color/white"
            android:textSize="25dp" />
        <EditText
            android:id="@+id/number"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="number"
            android:textColor="@android:color/white"
            android:textSize="25dp" >
```



Note

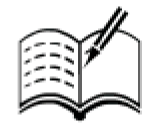
```
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <EditText
            android:id="@+id/message"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:hint="@string/message"
            android:inputType="textMultiLine"
            android:textColor="@android:color/white"
            android:textSize="25dp" />
    </LinearLayout>
    <Button
        android:id="@+id/send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button"
        android:textColor="@android:color/white"
        android:textSize="25dp" />
</LinearLayout>
```

（2）编写 SMSSenderActivity，通过为按钮增加单击事件监听器来完成发送短信功能。其代码如下：

```
public class SMSSenderActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                              //设置页面布局
        //通过 id 值获得文本框对象
        final EditText numberET = (EditText) findViewById(R.id.number);
        //通过 id 值获得文本框对象
        final EditText messageET = (EditText) findViewById(R.id.message);
        Button call = (Button) findViewById(R.id.send);             //通过 id 值获得按钮对象
        call.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String number = numberET.getText().toString();      //获得用户输入的号码
                String message = messageET.getText().toString();    //获得用户输入的短信
                Intent intent = new Intent();                       //创建 Intent 对象
                intent.setData(Uri.parse("smsto:" + number));       //设置要发送的号码
                intent.putExtra("sms_body", message);               //设置要发送的信息内容
                startActivity(intent);                              //将 Intent 传递给 Activity
            }
```



```
            });
        }
}
```



Note

（3）修改 AndroidManifest.xml 文件，增加发送短信的权限。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="17" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".SMSSenderActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

9.5.2 使用 Intent 打开网页

【例 9.6】 本实例使用 Intent 实现打开网页的功能，运行程序，运行效果如图 9.13 所示，单击“打开网页”按钮，显示如图 9.14 所示的谷歌主页。

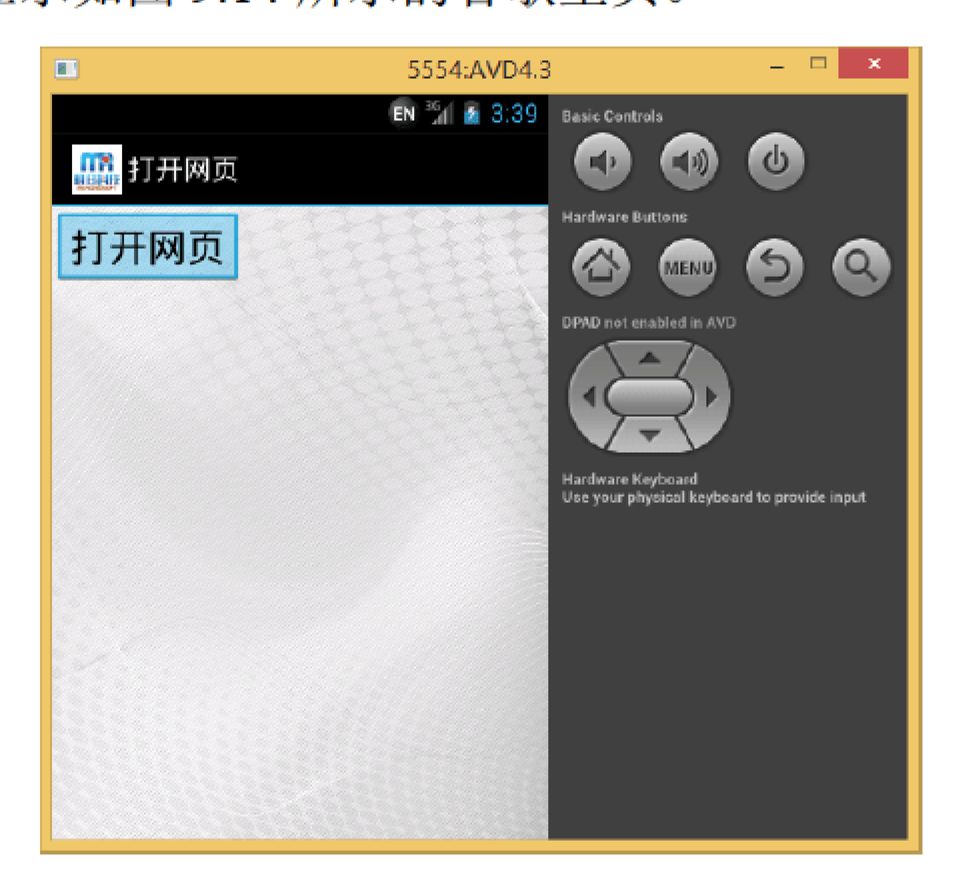


图 9.13　打开网页界面



Note

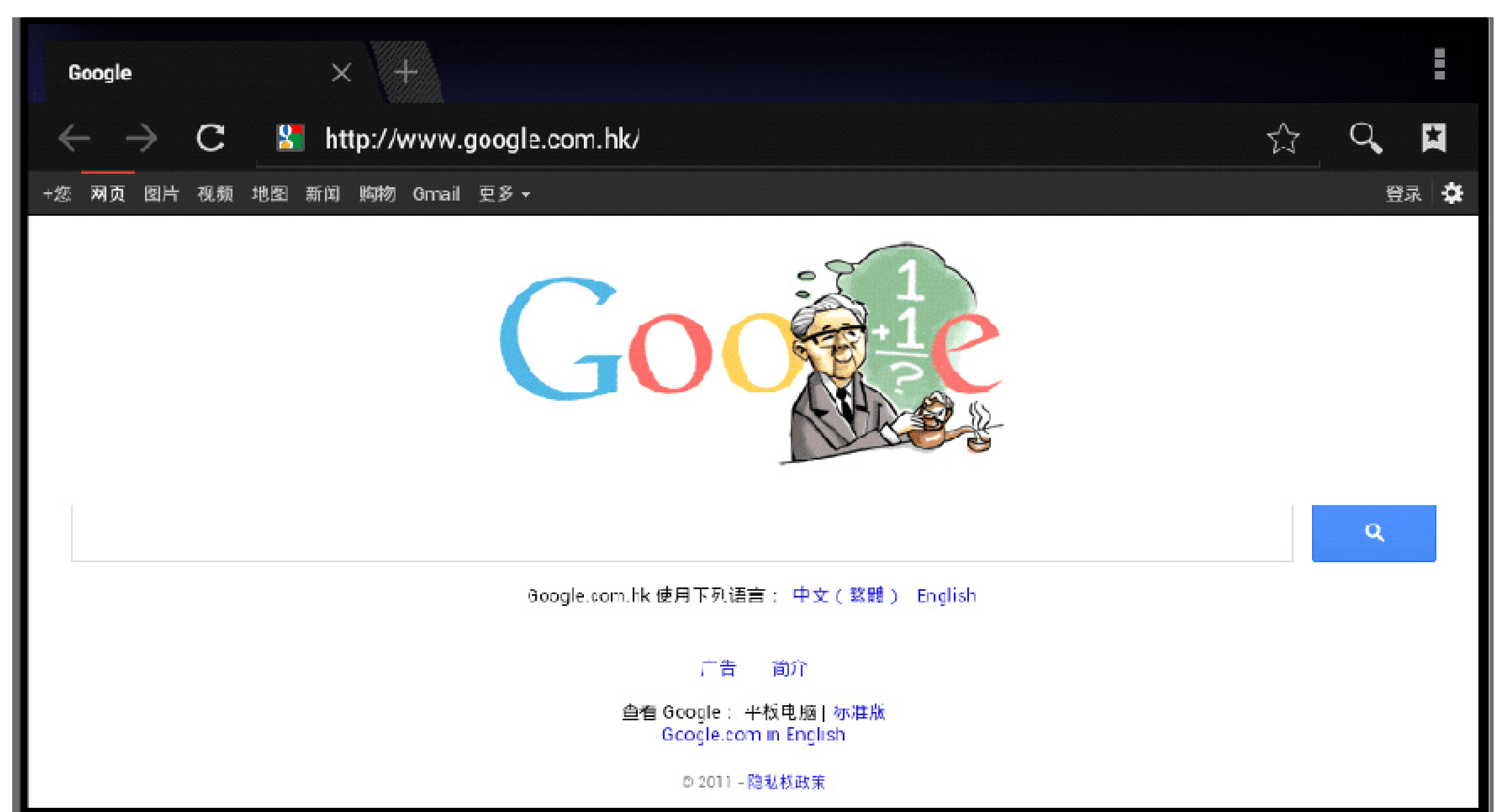


图 9.14　谷歌主页

实例位置：光盘\MR\Instance\09\9.6

> **说明：**
> 本实例默认打开的是谷歌的主页，用户可以根据自己的实际需求打开任何网页。

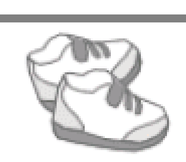
程序的开发步骤如下：

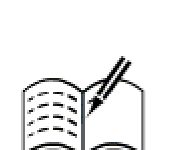
（1）在 res\layout 文件夹中打开布局文件 main.xml，增加一个按钮，并修改其默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/open"
        android:textColor="@android:color/black"
        android:textSize="25px" />
</LinearLayout>
```

（2）编写 WebActivity，它通过为按钮增加单击事件监听器来完成打开网页功能。其代码如下：

```
public class WebActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                          //设置页面布局
```





Note

```
            Button button = (Button) findViewById(R.id.button);              //通过 id 值获得按钮对象
            button.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                    Intent intent = new Intent();                            //创建 Intent 对象
                    intent.setAction(Intent.ACTION_VIEW);                    //为 Intent 设置动作
                    intent.setData(Uri.parse("http://www.google.com.hk"));   //为 Intent 设置数据
                    startActivity(intent);                                   //将 Intent 传递给 Activity
                }
            });
        }
}
```

（3）修改 AndroidManifest.xml 文件，增加要启动的 Activity。

9.6 本章常见错误

运行 Android 程序时，出现下面的错误提示。

```
android.util.AndroidRuntimeException:Calling startActivity() from outside of an Activity context requires
the FLAG_ACTIVITY_NEW_TASK flag. Is this really what you want?
```

上面的异常提示为“从外部的上下文调用了 startActivity()方法，需要设置为启动一个新的任务 FLAG_ACTIVITY_NEW_TASK”，根据该解释，只需要在.java 文件中添加下面一行代码即可解决该错误。

```
intent.setFlag(Intent.FLAG_ACTIVITY_NEW_TASK);
```

9.7 本 章 小 结

本章主要使用 Intent 通信进行了详细讲解，具体讲解过程中，首先对 Intent 对象及其 3 种传输机制进行了简单介绍，然后重点讲解了 Intent 对象的组成及如何使用 Intent 传递数据。学习本章内容时，读者需要重点掌握如何使用 Intent 传递数据，而在传递数据的过程中，会用到 Intent 对象的各个组成部分，所以读者对这部分知识也要熟练掌握。

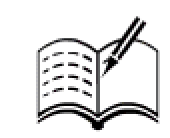
9.8 跟 我 上 机

参考答案：光盘\MR\跟我上机

创建一个 Android 程序，主要使用 Intent 实现返回系统 Home 桌面的功能。具体实现时，只

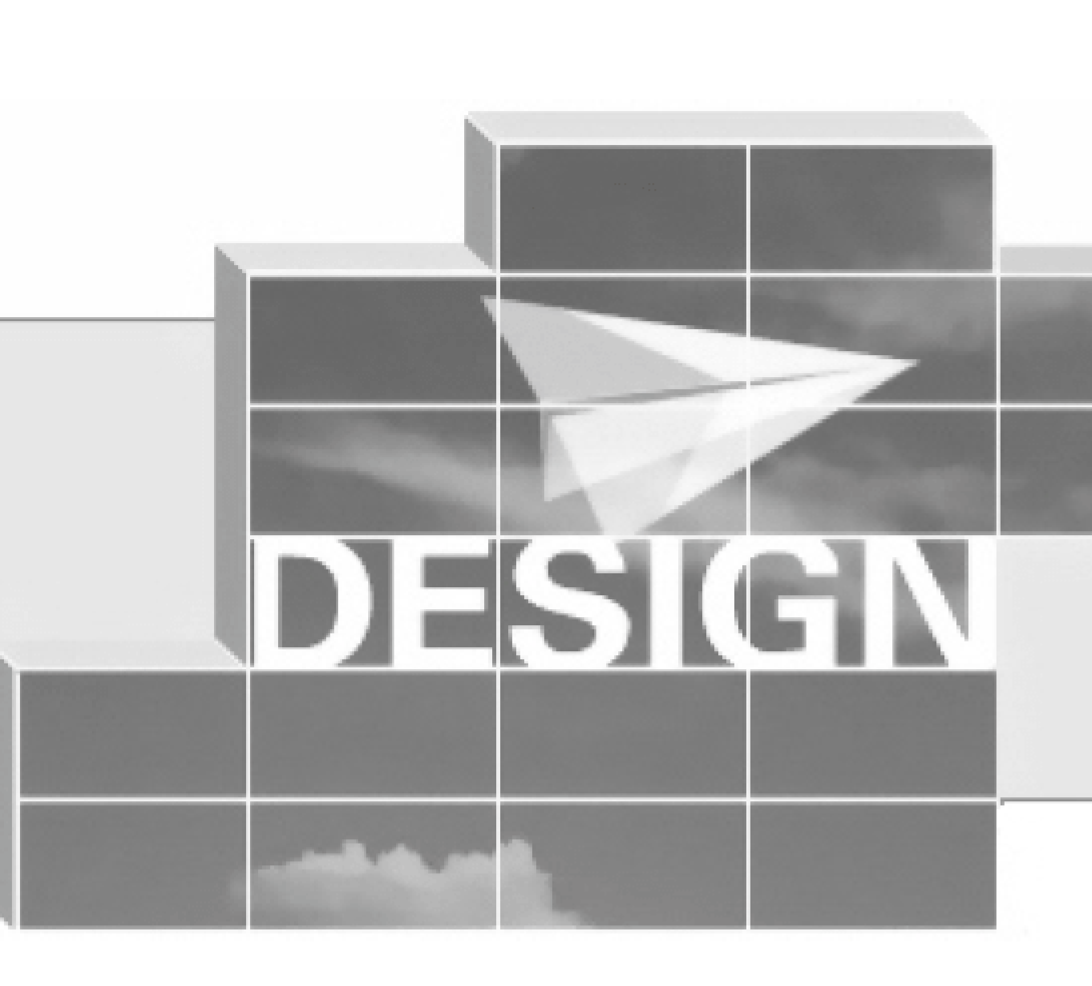


需要在布局文件中添加一个 Button 组件，然后在创建的 Activity 中获得布局文件中的按钮，并为其增加单击事件监听器，通过 Intent 对象的 setAction()和 addCategory()方法设置要返回到的界面，最后使用 startActivity()方法启动 Intent。其参考代码如下：




```
public class HomeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                                  //设置页面布局
        Button home = (Button) findViewById(R.id.home_button);          //通过 id 值获得按钮对象
        home.setOnClickListener(new View.OnClickListener() {            //为按钮增加单击事件监听器
            @Override
            public void onClick(View v) {
                Intent intent = new Intent();                           //创建 Intent 对象
                intent.setAction(Intent.ACTION_MAIN);                   //设置 Intent 动作
                intent.addCategory(Intent.CATEGORY_HOME);               //设置 Intent 种类
                startActivity(intent);//将 Intent 传递给 Activity
            }
        });
    }
}
```

第 2 篇



提高篇

- 第 10 章　Android 高级组件的使用
- 第 11 章　Android 中的事件处理
- 第 12 章　数据存储技术
- 第 13 章　Content Provider 实现数据共享
- 第 14 章　图形图像处理技术
- 第 15 章　利用 OpenGL 实现 3D 图形
- 第 16 章　多媒体应用开发
- 第 17 章　线程与消息处理
- 第 18 章　网络编程技术
- 第 19 章　Service 服务的使用



Android 高级组件的使用

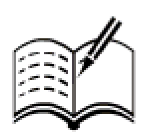
（视频讲解：1 小时 2 分钟）

经过前面的学习，已经可以设计出一些常用的 Android 界面，本章将继续学习 Android 开发中的用户界面设计，主要包括一些常用的高级组件、消息提示和对话框等，通过这些组件，可以开发出更加优秀的用户界面。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 在屏幕中显示模拟时钟
- ☐ 使用 DigitalClock 组件显示详细时间
- ☐ 在屏幕中显示水平进度条和圆形进度条
- ☐ 在屏幕中显示拖动条
- ☐ 在屏幕中显示星级评分条
- ☐ 显示消息提示框
- ☐ 在状态栏上显示通知
- ☐ 多种形式的列表对话框
- ☐ 显示在标题上的进度条
- ☐ 仿手机 QQ 登录状态显示功能





10.1　日期时间类组件

Android SDK 中提供了几种常用的日期时间类组件，主要包括 AnalogClock 和 DigitalClock 组件等，本节将分别对它们及其使用进行详细讲解。

10.1.1　AnalogClock 组件

AnalogClock 组件用来在 Android 中显示模拟时钟，它在显示时，只显示时针和分针。在 Android 中，如果想在屏幕中添加模拟时钟，可以在 XML 布局文件中通过<AnalogClock>标记添加。其基本语法格式如下：

```
<AnalogClock
    属性列表
/>
```

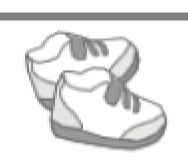
下面通过一个具体的实例来演示 AnalogClock 组件的使用。

【例 10.1】 在 Eclipse 中创建 Android 项目，主要演示如何使用 AnalogClock 组件在 Android 的 Activity 窗口中显示一个模拟时钟。

实例位置：光盘\MR\Instance\10\10.1

修改新建项目的 res\layout 目录下的布局文件 main.xml，将布局方式修改为 RelativeLayout，然后修改默认的 TextView 组件的文本、字体大小及颜色，最后添加一个 AnalogClock 组件，用来作为模拟时钟。main.xml 布局文件的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:textColor="#00ff00"
        android:text="AnalogClock 组件应用" />
    <AnalogClock
        android:id="@+id/analogClock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tv"
        android:layout_centerHorizontal="true"
```



```
        android:layout_marginTop="30dp"
        android:layerType="software" />
</RelativeLayout>
```



运行本实例，在屏幕中将显示如图 10.1 所示的模拟时钟。

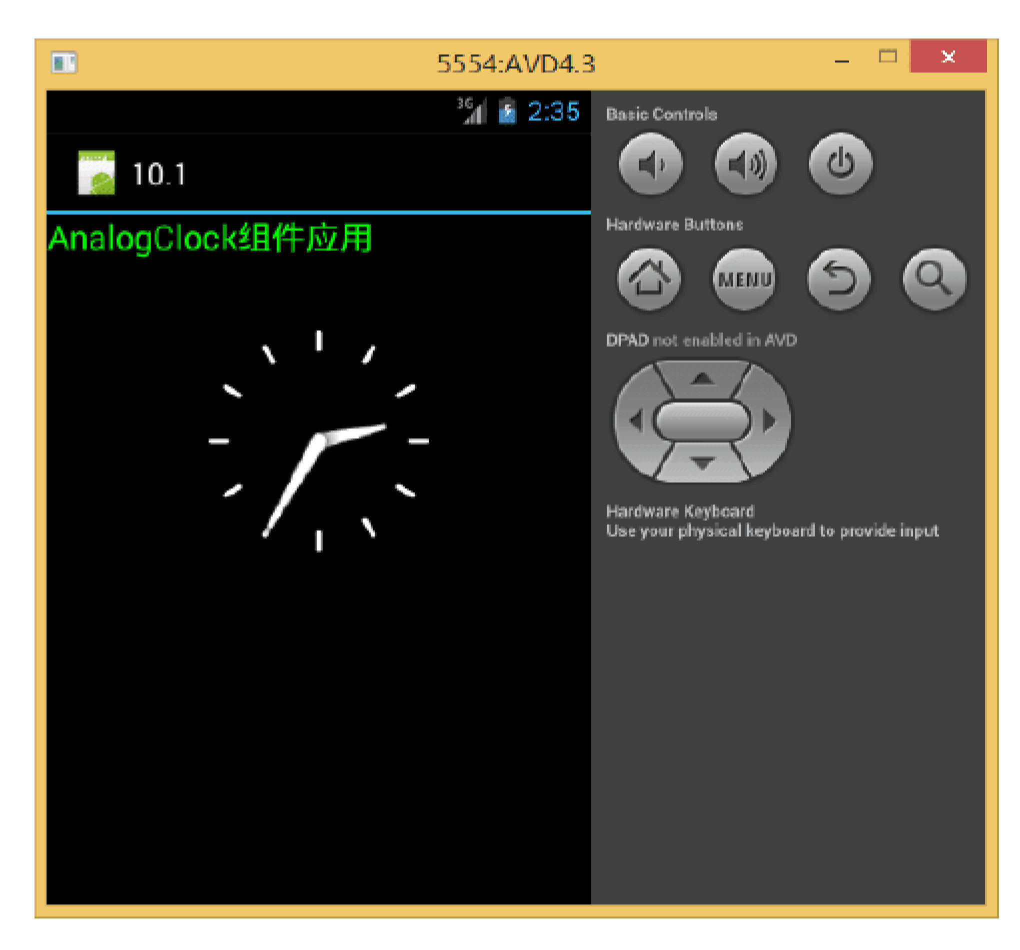


图 10.1　在屏幕中显示模拟时钟

10.1.2　DigitalClock 组件

DigitalClock 组件是一个用来显示详细时间的组件，在 Android 中，如果想在屏幕中添加该组件，可以在 XML 布局文件中通过<DigitalClock>标记添加。其基本语法格式如下：

```
<DigitalClock
    属性列表
/>
```

下面通过一个具体的实例来演示 DigitalClock 组件的使用。

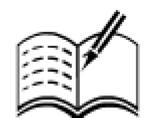
【例 10.2】 在 Eclipse 中创建 Android 项目，主要演示如何使用 DigitalClock 组件在 Android 的 Activity 窗口中显示当前的详细时间。

实例位置：光盘\MR\Instance\10\10.2

修改新建项目的 res\layout 目录下的布局文件 main.xml，将布局方式修改为 RelativeLayout，然后修改默认的 TextView 组件的文本、字体大小及颜色，最后添加一个 DigitalClock 组件，用来显示当前详细时间。main.xml 布局文件的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <TextView
```






```
            android:id="@+id/tv"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="20dp"
            android:textColor="#00ff00"
            android:text="DigitalClock 组件应用" />
        <DigitalClock
            android:id="@+id/digitalClock1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/tv"
            android:layout_centerHorizontal="true"
            android:layout_marginTop="32dp"
            android:text="DigitalClock"
            android:textSize="30dp"
            android:textColor="#ff0000" />
</RelativeLayout>
```

运行本实例，效果如图 10.2 所示。



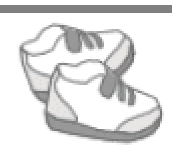
图 10.2　使用 DigitalClock 组件显示详细时间

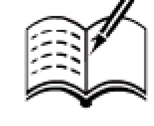
10.2　进度条组件

Android SDK 中提供了几种常用的进度条组件，主要包括 ProgressBar、SeekBar 和 RatingBar 组件等，本节将分别对它们及其使用进行详细讲解。

10.2.1　ProgressBar 组件

当一个应用在后台执行时，前台界面不会有任何信息，这时用户根本不知道程序是否在执行，





Note

以及执行进度等，这时就需要使用进度条来提示程序执行的进度。在 Android 中，进度条使用 ProgressBar 表示，用于向用户显示某个耗时操作完成的百分比。

在屏幕中添加进度条，可以在 XML 布局文件中通过<ProgressBar>标记添加，其基本语法格式如下：

```
< ProgressBar
      属性列表
>
</ ProgressBar>
```

ProgressBar 组件支持的 XML 属性如表 10.1 所示。

表 10.1　ProgressBar 组件支持的 XML 属性

XML 属性	描　　述
android:max	用于设置进度条的最大值
android:progress	用于指定进度条已完成的进度值
android:progressDrawable	用于设置进度条轨道的绘制形式

除了表 10.1 中介绍的属性，进度条组件还提供了下面两个常用方法用于操作进度。

☑ setProgress(int progress)方法：用于设置进度完成的百分比。

☑ incrementProgressBy(int diff)方法：用于设置进度条的进度增加或减少。当参数值为正数时表示进度增加，为负数时表示进度减少。

下面将给出一个关于在屏幕中使用进度条的实例。

【例 10.3】 在 Eclipse 中创建 Android 项目，实现水平进度条和圆形进度条。

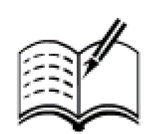
实例位置：光盘\MR\Instance\10\10.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并添加一个水平进度条和一个圆形进度条。修改后的代码如下：

```
<!-- 水平进度条 -->
<ProgressBar
      android:id="@+id/progressBar1"
      android:layout_width="match_parent"
      android:max="100"
      style="@android:style/Widget.ProgressBar.Horizontal"
      android:layout_height="wrap_content"/>
<!-- 圆形进度条 -->
<ProgressBar
      android:id="@+id/progressBar2"
      style="?android:attr/progressBarStyleLarge"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>
```







说明：

上面的代码中，通过 android:max 属性设置水平进度条的最大进度值；通过 style 属性可以为 ProgressBar 指定风格，常用的 style 属性值如表 10.2 所示。

表 10.2　ProgressBar 的 style 属性的可选值

style 属性	描　述
@android:attr/progressBarStyleHorizontal	细水平长条进度条
@android:attr/progressBarStyleLarge	大圆形进度条
@android:attr/progressBarStyleSmall	小圆形进度条
@android:style/Widget.ProgressBar.Large	大跳跃、旋转画面的进度条
@android:style/Widget.ProgressBar.Small	小跳跃、旋转画面的进度条
@android:style/Widget.ProgressBar.Horizontal	粗水平长条进度条

（2）在主活动 MainActivity 中，定义两个 ProgressBar 类的对象（分别用于表示水平进度条和圆形进度条）、一个 int 型的变量（用于表示完成进度）和一个处理消息的 Handler 类的对象。其代码如下：

```
private ProgressBar horizonP;                    //水平进度条
private ProgressBar circleP;                     //圆形进度条
private int mProgressStatus = 0;                 //完成进度
private Handler mHandler;                        //声明一个用于处理消息的 Handler 类的对象
```

（3）在主活动的 onCreate()方法中，首先获取水平进度条和圆形进度条，然后通过匿名内部类实例化处理消息的 Handler 类的对象，并重写其 handleMessage()方法，实现当耗时操作没有完成时，更新进度，否则设置进度条不显示。其关键代码如下：

```
horizonP = (ProgressBar) findViewById(R.id.progressBar1);      //获取水平进度条
circleP=(ProgressBar)findViewById(R.id.progressBar2);          //获取圆形进度条
mHandler=new Handler(){
    @Override
    public void handleMessage(Message msg) {
        if(msg.what==0x111){
            horizonP.setProgress(mProgressStatus);             //更新进度
        }else{
            Toast.makeText(MainActivity.this, "耗时操作已经完成", Toast.LENGTH_SHORT).show();
            horizonP.setVisibility(View.GONE);                 //设置进度条不显示，并且不占用空间
            circleP.setVisibility(View.GONE);                  //设置进度条不显示，并且不占用空间
        }
    }
};
```

（4）开启一个线程，用于模拟一个耗时操作。在该线程中，将调用 sendMessage()方法发送处理消息。其具体代码如下：



Note

```
new Thread(new Runnable() {
    public void run() {
        while (true) {
            mProgressStatus = doWork();                    //获取耗时操作完成的百分比
            Message m=new Message();
            if(mProgressStatus<100){
                m.what=0x111;
                mHandler.sendMessage(m);                   //发送信息
            }else{
                m.what=0x110;
                mHandler.sendMessage(m);                   //发送消息
                break;
            }
        }
    }
    //模拟一个耗时操作
    private int doWork() {
        mProgressStatus+=Math.random()*10;                 //改变完成进度
        try {
            Thread.sleep(200);                             //线程休眠 200 毫秒
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        return mProgressStatus;                            //返回新的进度
    }
}).start();                                                //开启一个线程
```

运行本实例，将显示如图 10.3 所示的运行效果。

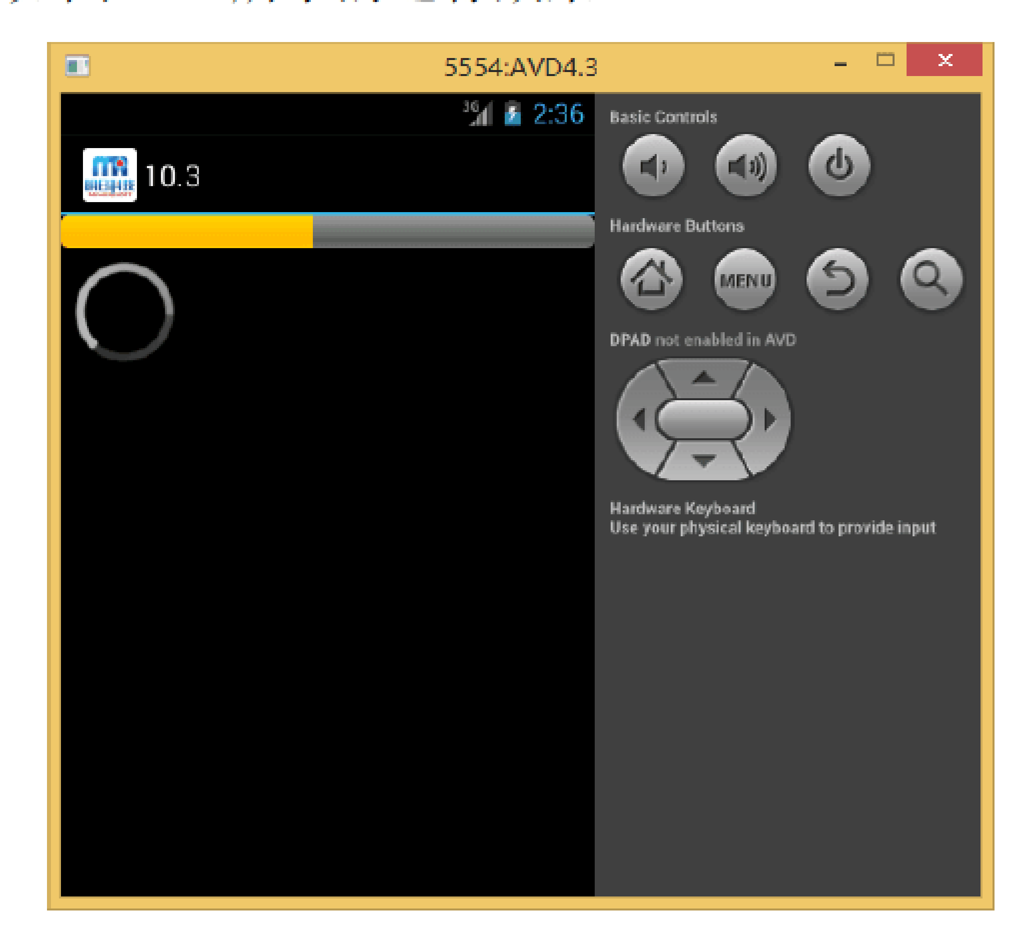
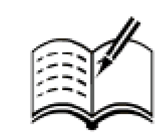


图 10.3　在屏幕中显示水平进度条和圆形进度条

10.2.2　SeekBar 组件

SeekBar 组件表示拖动条，它与进度条类似，所不同的是，拖动条允许用户拖动滑块来改变



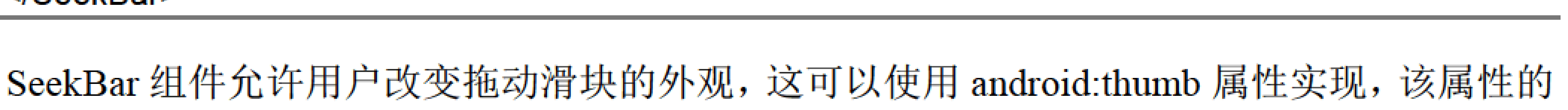


Note

值，通常用于实现对某种数值的调节如调节图片的透明度或是音量等。

在 Android 中，如果想在屏幕中添加拖动条，可以在 XML 布局文件中通过<SeekBar>标记添加。其基本语法格式如下：

```
<SeekBar
android:layout_height="wrap_content"
android:id="@+id/seekBar1"
android:layout_width="match_parent">
</SeekBar>
```



SeekBar 组件允许用户改变拖动滑块的外观，这可以使用 android:thumb 属性实现，该属性的属性值为一个 Drawable 对象，这个 Drawable 对象将作为自定义滑块。

由于拖动条可以被用户控制，所以需要为其添加 OnSeekBarChangeListener 监听器。为拖动条添加监听器的基本代码如下：

```
seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        //要执行的代码
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        //要执行的代码
    }
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,
            boolean fromUser) {
        //其他要执行的代码
    }
});
```

说明：

上面的代码中，onProgressChanged()方法中的参数 progress 表示当前进度，也就是拖动条的值。

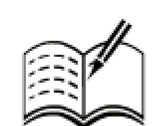
下面通过一个具体的实例来说明拖动条的具体应用。

【例 10.4】 在 Eclipse 中创建 Android 项目，实现在屏幕上显示拖动条，并为其添加 OnSeekBarChangeListener 监听器。

实例位置：光盘\MR\Instance\10\10.4

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件的 android:text 属性值修改为当前值：50，然后添加一个拖动条，并指定拖动条的当前值和最大


Note

值。修改后的代码如下：

```
<TextView
    android:text="当前值：50"
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<!-- 拖动条 -->
<SeekBar
    android:layout_height="wrap_content"
    android:id="@+id/seekBar1"
    android:max="100"
    android:progress="50"
    android:padding="10px"
    android:layout_width="match_parent"/>
```

（2）在主活动 MainActivity 中，定义一个 SeekBar 类的对象，用于表示拖动条。其具体代码如下：

```
private SeekBar seekbar;                                              //拖动条
```

（3）在主活动的 onCreate()方法中，首先获取布局文件中添加的文本视图和拖动条，然后为拖动条添加 OnSeekBarChangeListener 事件监听器，并且在重写的 onStopTrackingTouch()和 onStartTrackingTouch()方法中应用消息提示框显示对应状态，在 onProgressChanged()方法中修改文本视图的值为当前进度条的进度值。其具体代码如下：

```
final TextView result=(TextView)findViewById(R.id.textView1);        //获取文本视图
seekbar = (SeekBar) findViewById(R.id.seekBar1);                      //获取拖动条
seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {
        Toast.makeText(MainActivity.this, "结束滑动", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {
        Toast.makeText(MainActivity.this, "开始滑动", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser) {
        result.setText("当前值："+progress);                           //修改文本视图的值
    }
});
```

运行本实例，在屏幕中将显示默认进度为 50 的拖动条，如图 10.4 所示，用鼠标拖动圆形滑块，在上方的文本视图中将显示改变后的当前进度，并且通过消息提示框显示开始拖动和结束拖动。



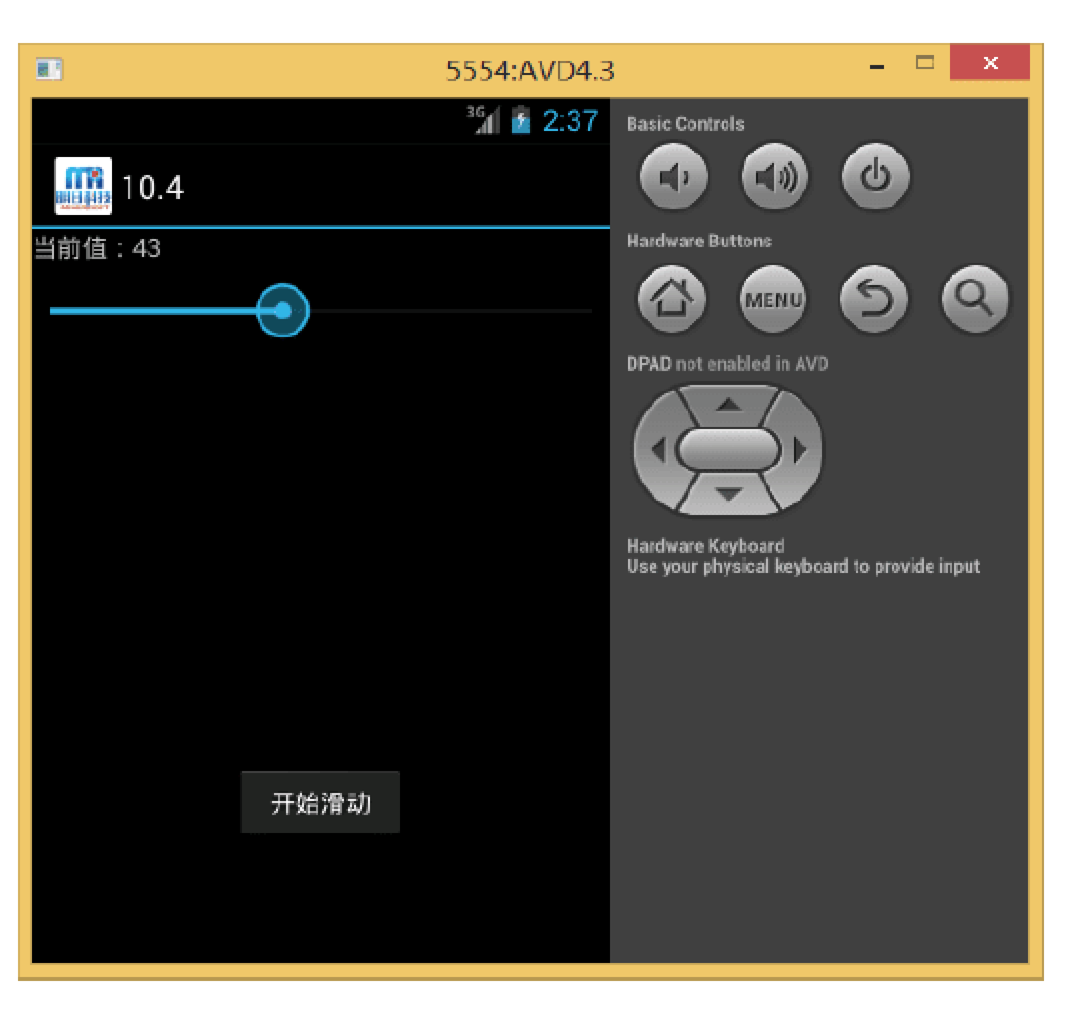


图 10.4　在屏幕中显示拖动条

Note

10.2.3　RatingBar 组件

星级评分条与进度条类似，都允许用户拖动来改变进度，所不同的是，星级评分条通过星星表示进度。通常情况，使用星级评分条表示对某一事物的支持度或是对某种服务的满意程度等。例如，淘宝网中对卖家的好评度，就是通过星级评分条实现的。

在 Android 中，如果想在屏幕中添加星级评分条，可以在 XML 布局文件中通过<RatingBar>标记添加。其基本语法格式如下：

```
<RatingBar
      属性列表
>
</RatingBar>
```

SeekBar 组件支持的 XML 属性如表 10.3 所示。

表 10.3　SeekBar 组件支持的 XML 属性

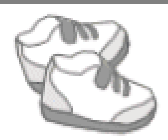
XML 属性	描　述
android:isIndicator	用于指定该星级评分条是否允许用户改变，true 为不允许改变
android:numStars	用于指定该星级评分条总共有多少个星
android:rating	用于指定该星级评分条默认的星级
android:stepSize	用于指定每次最少需要改变多少个星级，默认为 0.5 个

除了表 10.3 中介绍的属性外，星级评分条还提供了以下 3 个比较常用的方法。

☑　getRating()方法：用于获取等级，表示被选中了几颗星。

☑　getStepSize()：用于获取每次最少要改变多少个星级。

☑　getProgress()方法：用于获取进度，获取到的进度值等于 getRating()方法的返回值

*getStepSize()方法的返回值。

下面通过一个具体的实例来说明星级评分条的具体应用。

【例 10.5】 在 Eclipse 中创建 Android 项目，实现星级评分条。



实例位置：光盘\MR\Instance\10\10.5

Note

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并添加一个星级评分条和一个普通按钮。修改后的代码如下：

```
<!-- 星级评分条 -->
<RatingBar
        android:id="@+id/ratingBar1"
        android:numStars="5"
        android:rating="3.5"
        android:isIndicator="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
<Button
        android:text="提交"
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

（2）在主活动 MainActivity 中，定义一个 RatingBar 类的对象，用于表示星级评分条。其具体代码如下：

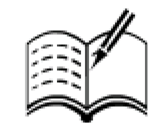
```
private RatingBar ratingbar;                                    //星级评分条
```

（3）在主活动的 onCreate()方法中，首先获取布局文件中添加的星级评分条，然后获取提交按钮，并为其添加单击事件监听器；在重写的 onClick()事件中获取进度、等级和每次最少要改变多少个星级，并显示到日志中，同时通过消息提示框显示获得的星的个数。其关键代码如下：

```
ratingbar = (RatingBar) findViewById(R.id.ratingBar1);          //获取星级评分条
Button button=(Button)findViewById(R.id.button1);               //获取提交按钮
button.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
               int result=ratingbar.getProgress();              //获取进度
               float rating=ratingbar.getRating();              //获取等级
               float step=ratingbar.getStepSize();              //获取每次最少要改变多少个星级
               Log.i("星级评分条","step="+step+" result="+result+" rating="+rating);
               Toast.makeText(MainActivity.this, "你得到了"+rating+"颗星", Toast.LENGTH_SHORT).show();
        }
});
```

运行本实例，在屏幕中将显示星级评分条，单击“提交”按钮，可以在弹出的消息提示框中显示有几颗星被选中，如图 10.5 所示。





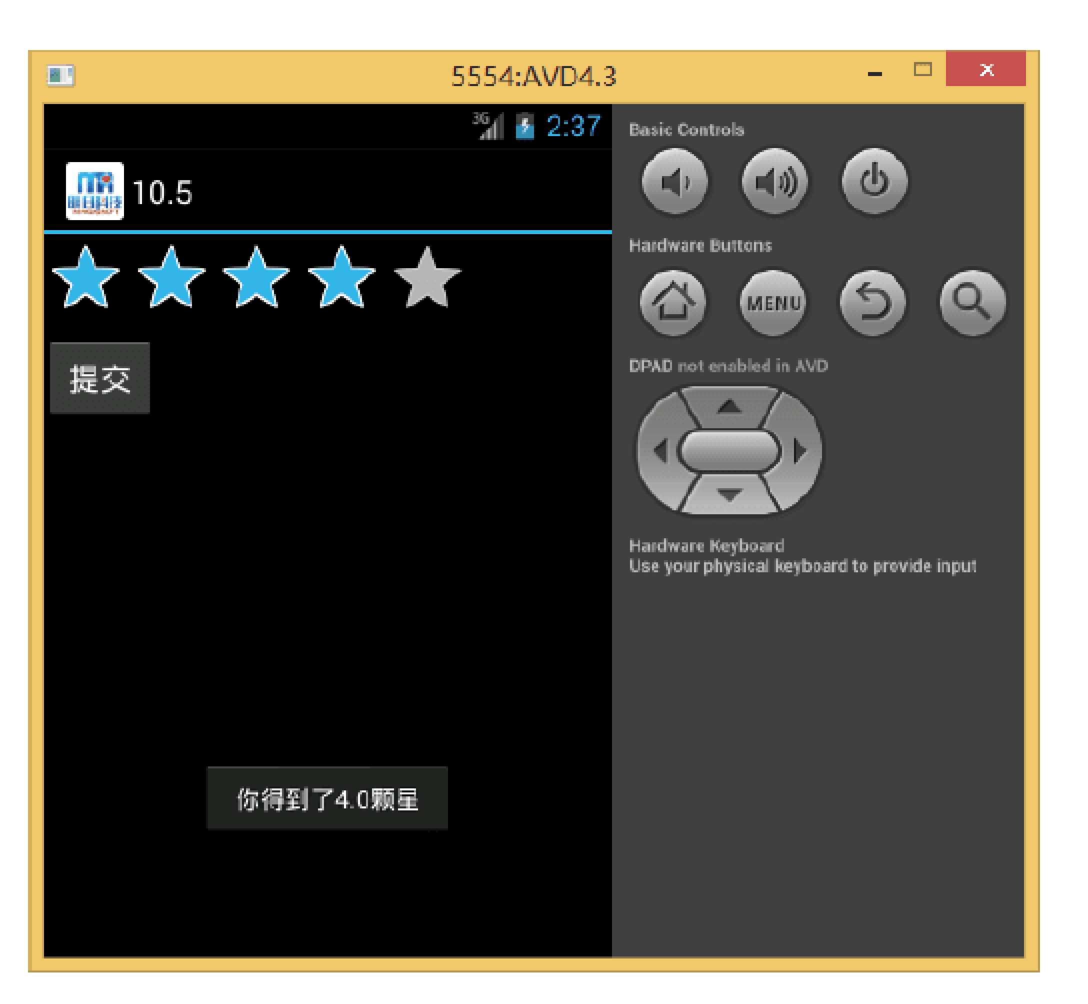


图 10.5　在屏幕中显示星级评分条

10.3　对话框及消息提示组件

在 Android 项目开发中，经常需要将一些临时信息显示给用户，虽然使用前面介绍的基本组件可以达到显示信息的目的。但是，这样做不仅会增加代码量，而且对于用户来说也不够友好。为此，Android 提供了消息提示框与对话框来显示这些信息。下面将分别介绍消息提示框与对话框的基本应用。

10.3.1　Toast 组件

Toast 类用于在屏幕中显示一个提示信息框，该消息提示框没有任何控制按钮，并且不会获得焦点，经过一定时间后自动消失。通常用于显示一些快速提示信息，应该范围非常广泛。

使用 Toast 来显示信消提示框，比较简单，只需要经过以下 3 个步骤即可实现。

（1）创建一个 Toast 对象。通常有两种方法，一种是使用构造方式进行创建，另一种是调用 Toast 类的 makeText()方法创建。

使用构造方法创建一个名称为 toast 的 Toast 对象的基本代码如下：

```
Toast toast=new Toast(this);
```

调用 Toast 类的 makeText()方法创建一个名称为 toast 的 Toast 对象的基本代码如下：

```
Toast toast=Toast.makeText(this, "要显示的内容", Toast.LENGTH_SHORT);
```

（2）调用 Toast 类提供的方法来设置该消息提示的对齐方式、页边距和显示的内容等。常用的方法如表 10.4 所示。

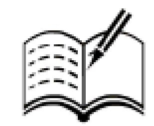

Note

表 10.4　Toast 类的常用方法

方　　法	描　　述
setDuration(int duration)	用于设置消息提示框持续的时间，通常使用 Toast.LENGTH_LONG 或 Toast.LENGTH_SHORT 参数值
setGravity(int gravity, int xOffset, int yOffset)	用于设置消息提示框的位置，参数 gravity 用于指定对齐方式，xOffset 和 yOffset 用于指定具体的偏移值
setMargin(float horizontalMargin, float verticalMargin)	用于设置消息提示的页边距
setText(CharSequence s)	用于设置要显示的文本内容
setView(View view)	用于设置将要在消息提示框中显示的视图

（3）调用 Toast 类的 show()方法显示消息提示框。需要注意的是，一定要调用该方法，否则设置的消息提示框将不显示。

下面通过一个具体的实例来说明如何使用 Toast 类显示消息提示框。

【例 10.6】在 Eclipse 中创建 Android 项目，通过两种方法显示消息提示框。

实例位置：光盘\MR\Instance\10\10.6

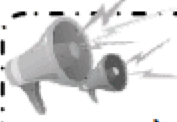
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，为默认添加的垂直线性布局设置一个 android:id 属性。其关键代码如下：

```
android:id="@+id/ll"
```

（2）在主活动 MainActivity.java 的 onCreate()方法中，通过 makeText()方法显示一个消息提示框。其关键代码如下：

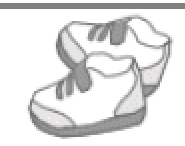
```
Toast.makeText(this, "我是通过 makeText()方法创建的消息提示框", Toast.LENGTH_LONG).show();
```


注意：

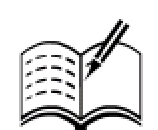
在最后一定不要忘记调用 show()方法，否则该消息提示框将不显示。

（3）通过 Toast 类的构造方法创建一个消息提示框，并设置该消息框的持续时间、对齐方式，以及要显示的内容等，这里设置其显示内容为带图标的消息。其具体代码如下：

```
Toast toast=new Toast(this);
toast.setDuration(Toast.LENGTH_SHORT);                //设置持续时间
toast.setGravity(Gravity.CENTER, 0, 0);               //设置对齐方式
LinearLayout ll=new LinearLayout(this);               //创建一个线性布局管理器
ImageView iv=new ImageView(this);                     //创建一个 ImageView
iv.setImageResource(R.drawable.alerm);                //设置要显示的图片
iv.setPadding(0, 0, 5, 0);                            //设置 ImageView 的右边距
ll.addView(iv);                                       //将 ImageView 添加到线性布局管理器中
TextView tv=new TextView(this);                       //创建一个 TextView
tv.setText("我是通过构造方法创建的消息提示框");        //为 TextView 设置文本内容
```



```
ll.addView(tv);                         //将 TextView 添加到线性布局管理器中
toast.setView(ll);                      //设置消息提示框中要显示的视图
toast.show();                           //显示消息提示框
```



Note

运行本实例，首先显示如图 10.6 所示的消息提示框，过一段时间后，该消息提示框消失，然后显示如图 10.7 所示的消息提示框，再过一段时间后，该消息提示框也自动消失。

我是通过makeText()方法创建的消息提示框

图 10.6　消息提示框一

我是通过构造方法创建的消息提示框

图 10.7　消息提示框二

10.3.2　Notification 组件

在使用手机时，当有未接来电或是新短消息时，手机会给出相应的提示信息，这些提示信息通常会显示到手机屏幕的状态栏上。Android 也提供了用于处理这些信息的类，它们是 Notification 和 NotificationManager。其中，Notification 代表的是具有全局效果的通知，而 NotificationManager 则是用于发送 Notification 通知的系统服务。

使用 Notification 和 NotificationManager 类发送和显示通知也比较简单，大致可以分为以下 4 个步骤。

（1）调用 getSystemService()方法获取系统的 NotificationManager 服务。

（2）创建一个 Notification 对象，并为其设置各种属性。

（3）为 Notification 对象设置事件信息。

（4）通过 NotificationManager 类的 notify()方法发送 Notification 通知。

下面通过一个具体的实例来说明如何使用 Notification 在状态栏上显示通知。

【例 10.7】 在 Eclipse 中创建 Android 项目，主要实现使用 Notification 在状态栏上显示通知的功能。

实例位置：光盘\MR\Instance\10\10.7

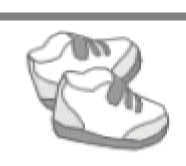
程序的开发步骤如下：

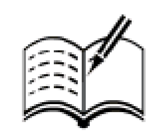
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加两个普通按钮，一个用于显示通知，另一个用户删除通知。由于此处的布局代码比较简单，这里就不再给出。

（2）在主活动 MainActivity.java 的 onCreate()方法中，调用 getSystemService()方法获取系统的 NotificationManager 服务。其关键代码如下：

```
//获取通知管理器，用于发送通知
final NotificationManager notificationManager =
            (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

（3）获取"显示通知"按钮，并为其添加单击事件监听器。在重写的 onClick()方法中，首先通过无参的构造方法创建一个 Notification 对象，并设置其相关属性，然后通过通知管理器发





Note

送该通知，接下来通过构造方法 Notification(int icon, CharSequence tickerText, long when)创建一个通知，并为其设置事件信息，最后通过通知管理器发送该通知。其具体代码如下：

```
Button button1 = (Button) findViewById(R.id.button1);              //获取“显示通知”按钮
//为“显示通知”按钮添加单击事件监听器
button1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        Notification notify = new Notification();                    //创建一个 Notification 对象
        notify.icon = R.drawable.advise;
        notify.tickerText = "显示第一个通知";
        notify.when = System.currentTimeMillis();                    //设置发送时间
        notify.defaults = Notification.DEFAULT_ALL;                  //设置默认声音、默认振动和默认闪光灯
        //设置事件信息
        notify.setLatestEventInfo(MainActivity.this, "无题", "每天进步一点点", null);
        notificationManager.notify(NOTIFYID_1, notify);              //通过通知管理器发送通知
        //添加第二个通知
        Notification notify1 = new Notification(R.drawable.advise2,
                "显示第二个通知", System.currentTimeMillis());
        notify1.flags|=Notification.FLAG_AUTO_CANCEL;                //打开应用程序后图标消失
        Intent intent=new Intent(MainActivity.this,ContentActivity.class);
        PendingIntent pendingIntent=PendingIntent.getActivity(MainActivity.this, 0, intent, 0);
        notify1.setLatestEventInfo(MainActivity.this, "通知",
                "查看详细内容", pendingIntent);                       //设置事件信息
        notificationManager.notify(NOTIFYID_2, notify1);             //通过通知管理器发送通知
    }
});
```

注意：

在上面代码中加粗的代码，为第一个通知设置使用默认声音、默认振动和默认闪光灯。也就是说，程序中需要访问系统闪光灯和振动器，这时就需要在 AndroidManifest.xml 中声明使用权限。其具体代码如下：

```
<!-- 添加操作闪光灯的权限 -->
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<!-- 添加操作振动器的权限 -->
<uses-permission android:name="android.permission.VIBRATE"/>
```

另外，在程序中还需要启动另一个活动 ContentActivity。因此，也需要在 AndroidManifest.xml 文件中声明该 Activity。其具体代码如下：

```
<activity android:name=".ContentActivity"
          android:label="详细内容"
          android:theme="@android:style/Theme.Dialog"/>
```

（4）获取“删除通知”按钮，并为其添加单击事件监听器，在重写的 onClick()方法中删除全部通知。其具体代码如下：




```
Button button2 = (Button) findViewById(R.id.button2);        //获取“删除通知”按钮
//为“删除通知”按钮添加单击事件监听器
button2.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
      //     notificationManager.cancel(NOTIFYID_1);        //清除 id 号为常量 NOTIFYID_1 的通知
             notificationManager.cancelAll();                 //清除全部通知
      }
});
```

（5）由于在为第二个通知指定事件信息时，为其关联了一个 Activity，因此，还需要创建该 Activity，在该 Activity 中，只需要通过一个 TextView 组件显示一行具体的通知信息。

运行本实例，单击“显示通知”按钮，在屏幕的左上角将显示第一个通知，如图 10.8 所示。过一段时间后，该通知消失，并显示第二个通知，再过一段时间后，该通知也消失，这时在状态栏上将显示这两个通知的图标，如图 10.9 所示。单击通知图标，将显示如图 10.10 所示的通知列表，单击第一个列表项，可以查看通知的详细内容，如图 10.11 所示。查看后，该通知的图标将不在状态栏中显示。单击“删除通知”按钮，可以删除全部通知。

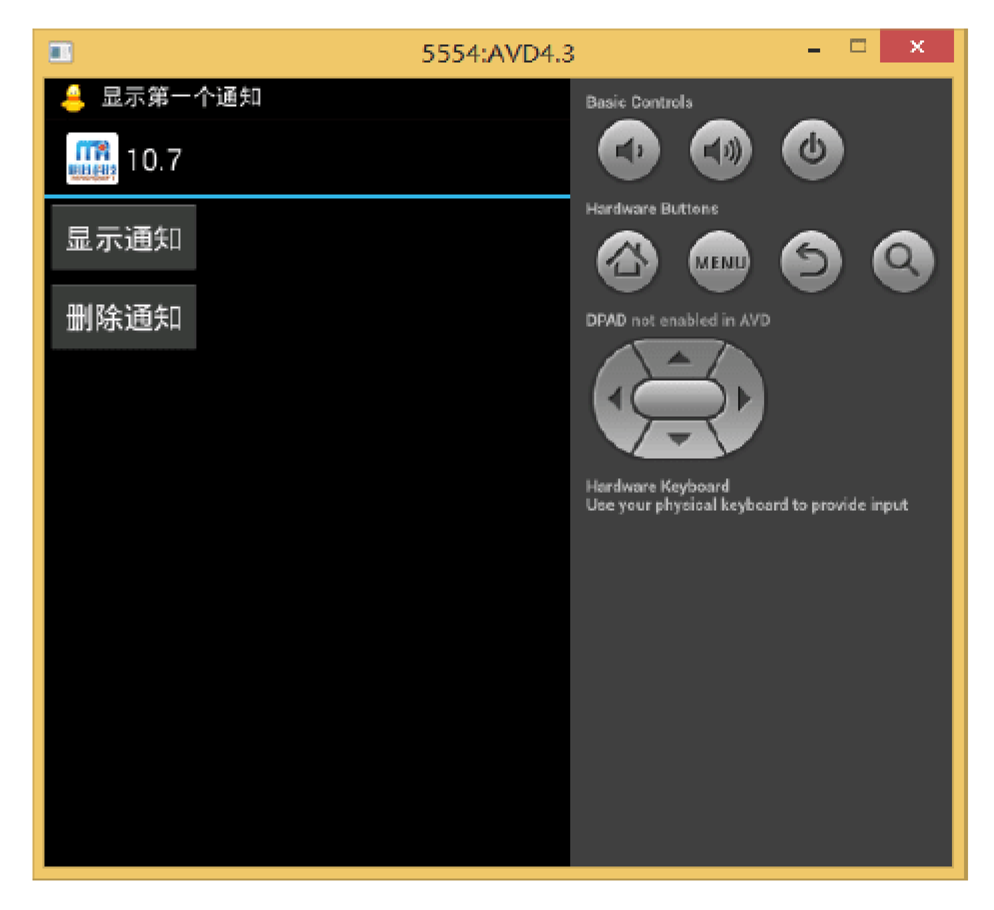


图 10.8　单击“显示通知”按钮后显示的通知

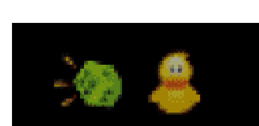


图 10.9　在状态栏上显示通知图标



图 10.10　单击状态栏上的通知图标显示通知列表



图 10.11　第一个通知的详细内容

10.3.3　AlertDialog 组件

AlertDialog 类的功能非常强大，它不仅可以生成带按钮的提示对话框，还可以生成带列表的列表对话框。使用 AlertDialog 可以生成的对话框概括起来有以下 4 种。

☑　带“确定”、“中立”和“取消”等 N 个按钮的提示对话框，其中的按钮个数不是固定



的，可以根据需要添加。例如，不需要有“中立”按钮，那么就可以生成只带有“确定”和“取消”按钮的对话框，也可以是只带有一个按钮的对话框。



Note

☑ 带列表的列表对话框。

☑ 带多个单选列表项和 N 个按钮的列表对话框。

☑ 带多个多选列表项和 N 个按钮的列表对话框。

在使用 AlertDialog 类生成对话框时，常用的方法如表 10.5 所示。

表 10.5　AlertDialog 类的常用方法

方　法	描　述
setTitle(CharSequence title)	用于为对话框设置标题
setIcon(Drawable icon)	用于为对话框设置图标
setIcon(int resId)	用于为对话框设置图标
setMessage(CharSequence message)	用于为提示对话框设置要显示的内容
setButton()	用于为提示对话框添加按钮，可以是“取消”、“中立”和“确定”按钮。需要通过为其指定 int 类型的 whichButton 参数实现，其参数值可以是 DialogInterface.BUTTON_POSITIVE（“确定”按钮）、BUTTON_NEGATIVE（“取消”按钮）或者 BUTTON_NEUTRAL（“中立”按钮）

通常情况下，使用 AlertDialog 类只能生成带 N 个按钮的提示对话框，要生成另外 3 种列表对话框，需要使用 AlertDialog.Builder 类。AlertDialog.Builder 类提供的常用方法如表 10.6 所示。

表 10.6　AlertDialog.Builder 类的常用方法

方　法	描　述
setTitle(CharSequence title)	用于为对话框设置标题
setIcon(Drawable icon)	用于为对话框设置图标
setIcon(int resId)	用于为对话框设置图标
setMessage(CharSequence message)	用于为提示对话框设置要显示的内容
setNegativeButton()	用于为对话框添加“取消”按钮
setPositiveButton()	用于为对话框添加“确定”按钮
setNeutralButton()	用于为对话框添加“中立”按钮
setItems()	用于为对话框添加列表项
setSingleChoiceItems()	用于为对话框添加单选列表项
setMultiChoiceItems()	用于为对话框添加多选列表项

下面通过一个具体的实例来说明如何应用 AlertDialog 类生成各种提示对话框和列表对话框。

【例 10.8】 在 Eclipse 中创建 Android 项目，应用 AlertDialog 类创建带“取消”、“中立”和“确定”按钮的提示对话框、带列表的列表对话框、带多个单选列表项的列表对话框和带多个多选列表项的列表对话框。

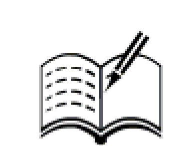
实例位置：光盘\MR\Instance\10\10.8

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件



删除，然后添加 4 个用于控制各种对话框显示的按钮。

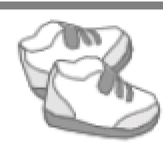


（2）在主活动 MainActivity.java 的 onCreate()方法中，获取布局文件中添加的第一个按钮，也就是“显示带取消、中立和确定按钮的对话框”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，应用 AlertDialog 类创建一个带“取消”、“中立”和“确定”按钮的提示对话框。其具体代码如下：

```
//获取“显示带取消、中立和确定按钮的对话框”按钮
Button button1 = (Button) findViewById(R.id.button1);
//为“显示带取消、中立和确定按钮的对话框”按钮添加单击事件监听器
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        AlertDialog alert = new AlertDialog.Builder(MainActivity.this).create();
        alert.setIcon(R.drawable.advise);                          //设置对话框的图标
        alert.setTitle("系统提示：");                              //设置对话框的标题
        alert.setMessage("带取消、中立和确定按钮的对话框！");      //设置要显示的内容
        //添加“取消”按钮
        alert.setButton(DialogInterface.BUTTON_NEGATIVE,"取消", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您单击了取消按钮",Toast.LENGTH_SHORT).show();
            }
        });
        //添加“确定”按钮
        alert.setButton(DialogInterface.BUTTON_POSITIVE,"确定", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "您单击了确定按钮",Toast.LENGTH_SHORT).show();
            }
        });
        alert.setButton(DialogInterface.BUTTON_NEUTRAL,"中立",new OnClickListener(){
            @Override
            public void onClick(DialogInterface dialog, int which) {}
        });                                                        //添加“中立”按钮
        alert.show();                                              //显示对话框
    }
});
```

（3）在主活动 MainActivity.java 的 onCreate()方法中，获取布局文件中添加的第二个按钮，也就是“显示带列表的对话框”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，应用 AlertDialog 类创建一个带 5 个列表项的列表对话框。其具体代码如下：

```
Button button2 = (Button) findViewById(R.id.button2);          //获取“显示带列表的对话框”按钮
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String[] items = new String[] { "跑步", "羽毛球", "乒乓球", "网球", "体操" };
        Builder builder = new AlertDialog.Builder(MainActivity.this);
```

```
            builder.setIcon(R.drawable.advise1);                       //设置对话框的图标
            builder.setTitle("请选择你喜欢的运动项目：");               //设置对话框的标题
            //添加列表项
            builder.setItems(items, new OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Toast.makeText(MainActivity.this,
                            "您选择了" + items[which], Toast.LENGTH_SHORT).show();
                }
            });
            builder.create().show();                                   //创建对话框并显示
        }
});
```



注意：

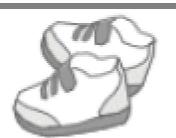
上面代码中加粗的代码，一定不要忘记，否则将不能显示生成的对话。

（4）在主活动 MainActivity.java 的 onCreate()方法中，获取布局文件中添加的第 3 个按钮，也就是“显示带单选列表项的对话框”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，应用 AlertDialog 类创建一个带 5 个单选列表项和 1 个“确定”按钮的列表对话框。其具体代码如下：

```
Button button3 = (Button) findViewById(R.id.button3);            //获取“显示带单选列表项的对话框”按钮
button3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final String[] items = new String[] { "标准", "无声", "会议", "户外","离线" };
            //显示带单选列表项的对话框
            Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setIcon(R.drawable.advise2);                  //设置对话框的图标
            builder.setTitle("请选择要使用的情景模式：");          //设置对话框的标题
            builder.setSingleChoiceItems(items, 0, new OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    Toast.makeText(MainActivity.this,
                            //显示选择结果
                            "您选择了" + items[which], Toast.LENGTH_SHORT).show();
                }
            });

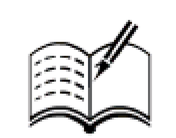
            builder.setPositiveButton("确定", null);               //添加“确定”按钮
            builder.create().show();                              //创建对话框并显示
        }
});
```

（5）在主活动中定义一个 boolean 类型的数组（用于记录各列表项的状态）和一个 String 类型的数组（用于记录各列表项要显示的内容）。其关键代码如下：



```
private boolean[] checkedItems;                          //记录各列表项的状态
private String[] items;                                  //各列表项要显示的内容
```

（6）在主活动 MainActivity.java 的 onCreate()方法中，获取布局文件中添加的第 4 个按钮，也就是“显示带多选列表项的对话框”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，应用 AlertDialog 类创建一个带 5 个多选列表项和 1 个“确定”按钮的列表对话框。其具体代码如下：

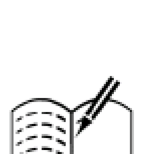


Note

```
Button button4 = (Button) findViewById(R.id.button4);          //获取“显示带多选列表项的对话框”按钮
button4.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        checkedItems= new boolean[] { false, true, false,true, false };//各列表项的状态
        //各列表项要显示的内容
        items = new String[] { "植物大战僵尸", "愤怒的小鸟", "泡泡龙", "开心农场", "超级玛丽" };
        //显示带单选列表项的对话框
        Builder builder = new AlertDialog.Builder(MainActivity.this);
        builder.setIcon(R.drawable.advise2);                     //设置对话框的图标
        builder.setTitle("请选择您喜爱的游戏：");                  //设置对话框标题
        builder.setMultiChoiceItems(items, checkedItems,
                new OnMultiChoiceClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,int which, boolean isChecked) {
                        checkedItems[which]=isChecked;     //改变被操作列表项的状态
                    }
                });
        //为对话框添加“确定”按钮
        builder.setPositiveButton("确定", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String result="";                                //用于保存选择结果
                for(int i=0;i<checkedItems.length;i++){
                    if(checkedItems[i]){                         //当选项被选择时
                        result+=items[i]+"、";                   //将选项的内容添加到 result 中
                    }
                }
                //当 result 不为空时，通过消息提示框显示选择的结果
                if(!"".equals(result)){
                result=result.substring(0, result.length()-1);   //去掉最后面添加的“、”号
                    Toast.makeText(MainActivity.this,
                            "您选择了[ "+result+" ]", Toast.LENGTH_LONG).show();
                }
            }
        });
        builder.create().show();                                 //创建对话框并显示
    }
});
```

运行本实例，在屏幕中将显示 4 个按钮，单击第 1 个按钮，将弹出带“取消”、“中立”和“确





Note

定”按钮的对话框，如图 10.12 所示；单击第 2 个按钮，将弹出如图 10.13 所示的带列表的对话框，单击任何一个列表项，都将关闭该对话框，并显示一个消息提示框显示选取的内容；单击第 3 个按钮，将显示如图 10.14 所示的列表对话框，单击“确定”按钮，关闭该对话框；单击第 4 个按钮，将显示一个如图 10.15 所示的带 5 个多选列表项和 1 个“确定”按钮的列表对话框，选中多个列表项后，单击“确定”按钮，将显示如图 10.16 所示的消息提示框显示选取的内容。



图 10.12　带“取消”、“中立”和“确定”按钮的对话框



图 10.13　带列表的列表对话框



图 10.14　带单选列表的列表对话框



图 10.15　带多选列表的列表对话框

您选择了[愤怒的小鸟、泡泡龙、开心农场]

图 10.16　消息提示框

10.4　综 合 应 用

10.4.1　显示在标题上的进度条

【例 10.9】 本实例使用 ProgressBar 组件制作一个在标题上显示的进度条，运行程序，首先在标题上显示载入进度条，载入完毕后，显示载入后的 4 张图片，效果如图 10.17 所示。



Note

实例位置：光盘\MR\Instance\10\10.9

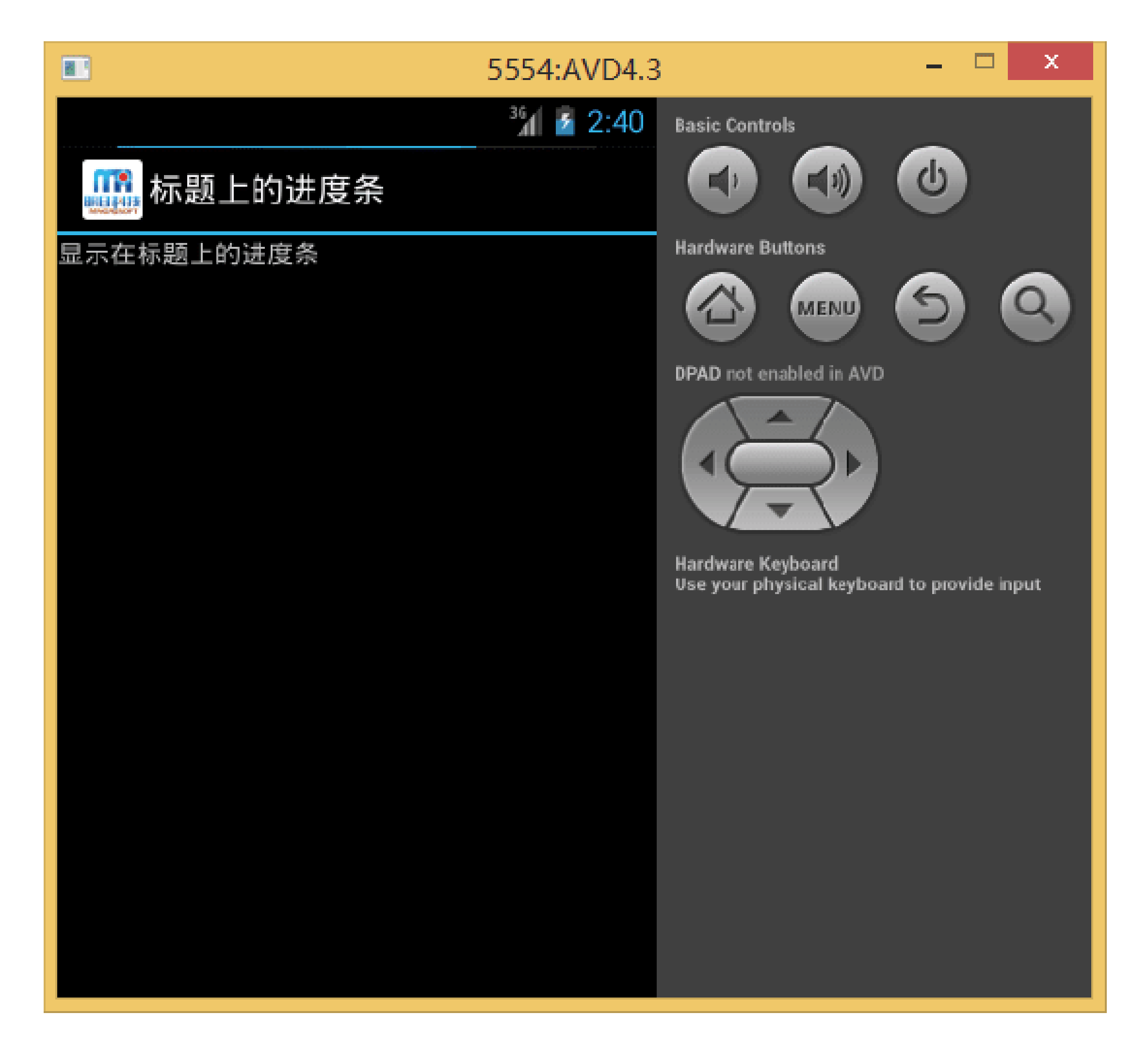




图 10.17　在标题上显示载入进度条

本实例实现时，首先通过 setProgressBarVisibility()方法在标题上显示进度条，然后通过线程控制加载任务的执行，并通过重写 onProgressUpdate()方法动态更新进度条进度。程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，为默认添加的垂直线性布局管理器设置一个 android:id 属性。其关键代码如下：

```
android:id="@+id/linearlayout1"
```

（2）在主活动 MainActivity 中，定义一个用于保存要显示图片 id 的数组（需要将要显示的图片复制到 res\drawable 文件夹中）和一个垂直线性布局管理器的对象。其关键代码如下：

```
private int imageId[] = new int[] { R.drawable.img01, R.drawable.img02,
            R.drawable.img03, R.drawable.img04 };          //定义并初始化一个保存要显示图片 id 的数组
private LinearLayout l;                                     //定义一个垂直线性布局管理器的对象
```

（3）在主活动的 onCreate()方法中，首先设置显示水平进度条，然后设置要显示的视图，这里为主布局文件 main.xml，接下来再获取布局文件中添加的垂直线性布局管理器。其关键代码如下：

```
requestWindowFeature(Window.FEATURE_PROGRESS); //显示水平进度条
setContentView(R.layout.main);
l = (LinearLayout) findViewById(R.id.linearlayout1);        //获取布局文件中添加的垂直线性布局管理器
```

（4）创建继承自 AsyncTask 的异步类，并重写 onPreExecute()、doInBackground()、onProgressUpdate()和 onPostExecute()方法，实现在向页面添加图片时，在标题上显示一个水平进度条，当图片载入完毕后，让进度条隐藏并显示图片。其具体代码如下：

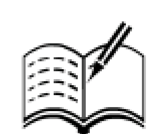


Note

```
/**
 * 功能：创建异步任务，添加 4 张图片
 *
 */
class MyTack extends AsyncTask<Void, Integer, LinearLayout> {
	@Override
	protected void onPreExecute() {
		setProgressBarVisibility(true);					//执行任务前让进度条可见
		super.onPreExecute();
	}
	/*
	 * 功能：要执行的耗时任务
	 */
	@Override
	protected LinearLayout doInBackground(Void... params) {
		LinearLayout ll = new LinearLayout(MainActivity.this);	//创建水平线性布局管理器
		for (int i = 1; i < 5; i++) {
			ImageView iv = new ImageView(MainActivity.this);	//创建 ImageView 对象
			iv.setLayoutParams(new LayoutParams(245, 108));
			iv.setImageResource(imageId[i - 1]);			//设置要显示的图片
			ll.addView(iv);				//将 ImageView 添加到线性布局管理器中
			try {
				Thread.sleep(10);			//为了更好地看到效果，这里让线程休眠 10 毫秒
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
			publishProgress(i);			//触发 onProgressUpdate(Progress...)方法更新进度
		}
		return ll;
	}
	/*
	 * 功能：更新进度
	 */
	@Override
	protected void onProgressUpdate(Integer... values) {
		setProgress(values[0] * 2500);			//动态更新最新进度
		super.onProgressUpdate(values);
	}
	/*
	 * 功能：任务执行后
	 */
	@Override
	protected void onPostExecute(LinearLayout result) {
		setProgressBarVisibility(false);			//任务执行后让进度条隐藏
		l.addView(result);	//将水平线性布局管理器添加到布局文件中添加的垂直线性布局管理器中
		super.onPostExecute(result);
	}
}
```

（5）在 onCreate()方法的最后执行自定义的任务 MyTack，其具体代码如下：






```
new MyTack().execute();                                   //执行自定义任务
```

10.4.2　仿手机 QQ 登录状态显示功能

【例 10.10】 本实例将实现仿手机 QQ 登录状态显示的功能，运行程序，将显示一个用户登录界面，如图 10.18 所示，输入用户名和密码后。单击“登录”按钮，将弹出如图 10.19 所示的选择登录状态的列表对话框，单击代表登录状态的列表项，该对话框消失，并在屏幕的左上角显示代表登录状态的通知，过一段时间后该通知消失，同时在状态栏上显示代表登录状态的图标，向下滑动该图标，将显示通知列表，如图 10.20 所示。单击“退出”按钮，可以删除该通知。



图 10.18　登录界面

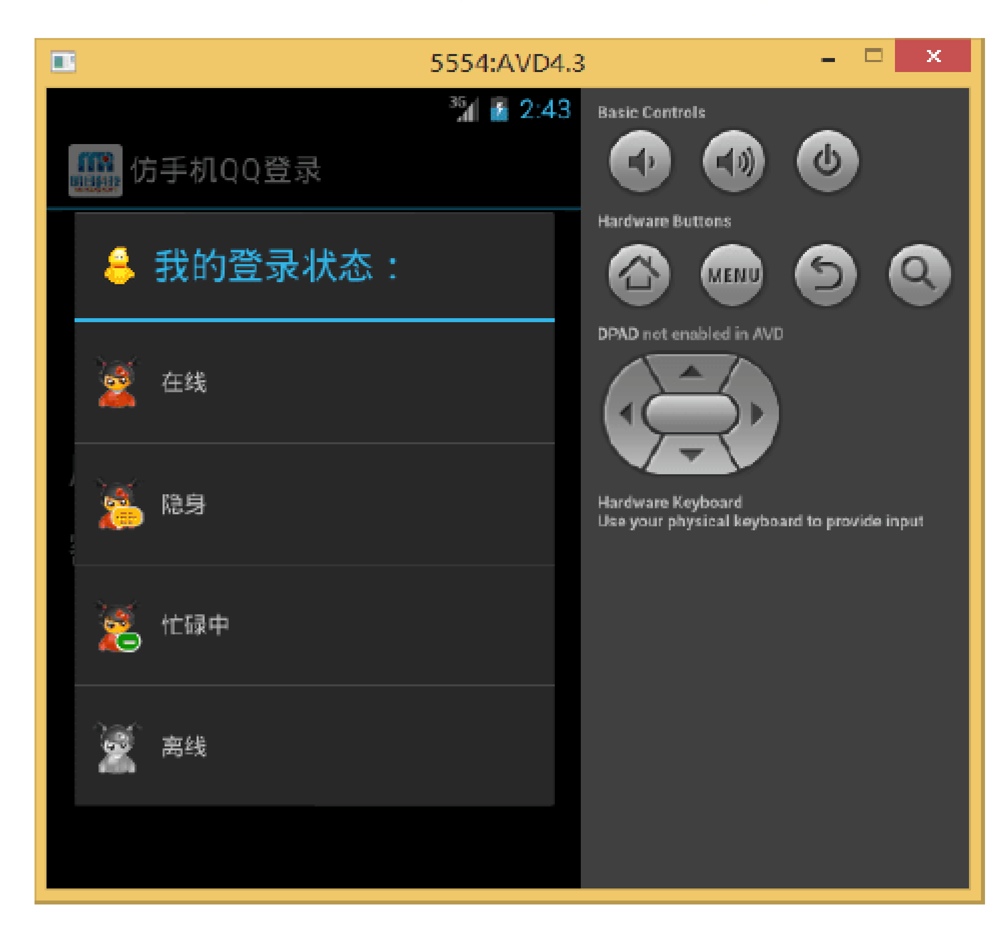


图 10.19　弹出的选择登录状态的列表对话框

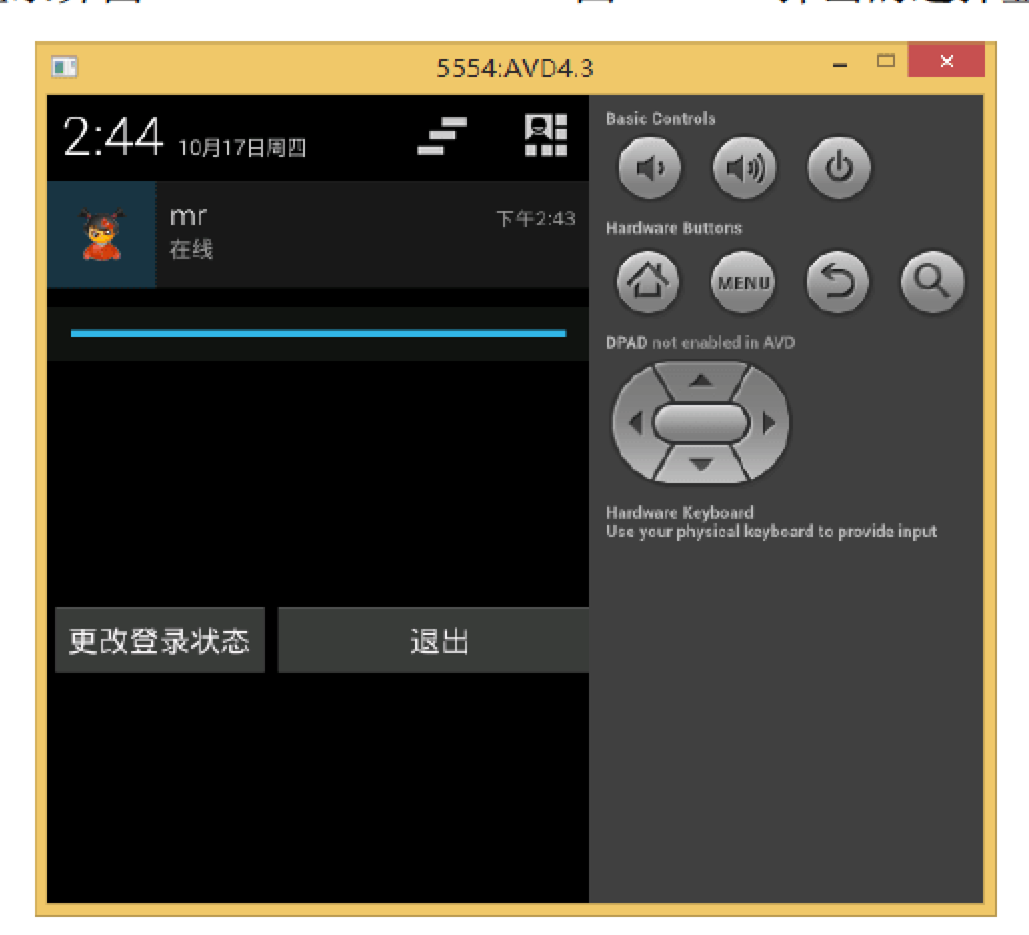
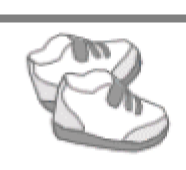


图 10.20　在状态栏中显示登录状态

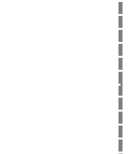
实例位置：光盘\MR\Instance\10\10.10

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，



然后添加一个 TableLayout 表格布局管理器，并且在该布局管理器中添加 3 个 TableRow 表格行，接下来再在每个表格行中添加用户登录界面相关的组件，最后设置表格的第 1 列和第 4 列允许被拉伸。



Note

（2）在主活动中，定义一个整型的常量（记录通知的 id）、一个 String 类型的变量（记录用户名）和一个通知管理器对象。其关键代码如下：

```
final int NOTIFYID_1 = 123;                                    //第一个通知的 id
private String user="匿名";                                    //用户名
private NotificationManager notificationManager;               //定义通知管理器对象
```

（3）在主活动的 onCreate()方法中，首先获取通知管理器，然后获取“登录”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中获取输入的用户名并调用自定义方法 sendNotification()发送通知。其具体代码如下：

```
//获取通知管理器，用于发送通知
notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
Button button1 = (Button) findViewById(R.id.button1);          //获取“登录”按钮
//为“登录”按钮添加单击事件监听器
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText etUser=(EditText)findViewById(R.id.user);     //获取“用户名”编辑框
        if(!"".equals(etUser.getText())){
            user=etUser.getText().toString();
        }
        sendNotification();                                    //发送通知
    }
});
```

（4）编写 sendNotification()方法，在该方法中，首先创建一个 AlertDialog.Builder 对象，并为其指定要显示对话框的图标、标题等，然后创建两个用于保存列表项图片 id 和文字的数组，并将这些图片 id 和文字添加到 List 集合中，再创建一个 SimpleAdapter 简单适配器，并将该适配器作为 Builder 对象的适配器用于为列表对话框添加带图标的列表项，最后创建对话框并显示。sendNotification()方法的具体代码如下：

```
//发送通知
private void sendNotification() {
    Builder builder = new AlertDialog.Builder(MainActivity.this);
    builder.setIcon(R.drawable.advise);                        //设置对话框的图标
    builder.setTitle("我的登录状态：");                          //设置对话框的标题
    final int[] imageId = new int[] { R.drawable.img1, R.drawable.img2,
            R.drawable.img3, R.drawable.img4 };                //定义并初始化保存图片 id 的数组
    //定义并初始化保存列表项文字的数组
    final String[] title = new String[] { "在线", "隐身", "忙碌中", "离线" };
    //创建一个 list 集合
    List<Map<String, Object>> listItems = new ArrayList<Map<String, Object>>();
    //通过 for 循环将图片 id 和列表项文字放到 Map 中，并添加到 list 集合中
```





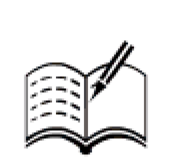

```
    for (int i = 0; i < imageId.length; i++) {
        Map<String, Object> map = new HashMap<String, Object>();   //实例化 Map 对象
        map.put("image", imageId[i]);
        map.put("title", title[i]);
        listItems.add(map);                                         //将 map 对象添加到 List 集合中
    }
    final SimpleAdapter adapter = new SimpleAdapter(MainActivity.this,
            listItems, R.layout.items, new String[] { "title", "image" },
            new int[] { R.id.title, R.id.image });                  //创建 SimpleAdapter
    builder.setAdapter(adapter, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Notification notify = new Notification();               //创建一个 Notification 对象
            notify.icon = imageId[which];
            notify.tickerText = title[which];
            notify.when = System.currentTimeMillis();               //设置发送时间
            notify.defaults = Notification.DEFAULT_SOUND;           //设置默认声音
            notify.setLatestEventInfo(MainActivity.this, user,
                    title[which], null);                            //设置事件信息
            notificationManager.notify(NOTIFYID_1, notify);         //通过通知管理器发送通知
            //让布局中的第一行不显示
            ((TableRow)findViewById(R.id.tableRow1)).setVisibility(View.INVISIBLE);
            //让布局中的第二行不显示
            ((TableRow)findViewById(R.id.tableRow2)).setVisibility(View.INVISIBLE);
            //改变"登录"按钮上显示的文字
            ((Button)findViewById(R.id.button1)).setText("更改登录状态");
        }
    });
    builder.create().show();                                        //创建对话框并显示
}
```

说明：

当用户选择了登录状态列表项后，在显示通知的同时，还需要将布局中的第一行（用于输入用户名）和第二行（用于输入密码）的内容设置为不显示，并且改变“登录”按钮上显示的文字为“更改登录状态”。

（5）在 onCreate()方法中，获取“退出”按钮，并为其添加单击事件监听器；在重写的 onClick()方法中，清除代表登录状态的通知，然后将布局中的第一行和第二行的内容显示出来，并改变“更改登录状态”按钮上显示的文字为“登录”。其具体代码如下：

```
Button button2 = (Button) findViewById(R.id.button2);          //获取"退出"按钮
//为"退出"按钮添加单击事件监听器
button2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        notificationManager.cancel(NOTIFYID_1);                 //清除通知
        //让布局中的第一行显示
```


Note

```
            ((TableRow)findViewById(R.id.tableRow1)).setVisibility(View.VISIBLE);
            //让布局中的第二行显示
            ((TableRow)findViewById(R.id.tableRow2)).setVisibility(View.VISIBLE);
            //改变“更改登录状态”按钮上显示的文字
            ((Button)findViewById(R.id.button1)).setText("登录");
        }
});
```

10.5 本章常见错误

运行 10.4.1 节的“显示在标题上的进度条”实例时，在 Android 模拟器上只显示第一个界面，然后程序就停止运行，看不到进度条加载完出现的图片，并且有下面的异常提示：

```
AndroidRuntime(2088):Caused by:java.lang.OutOfMemoryError
```

从异常提示分析是内存溢出异常，该异常是由于 Android 模拟器的内存设置比较小造成的，可以通过如下两种方法解决。

☑ 修改 Android 模拟器的内存。

☑ 将程序中的图片尺寸修改小一些。

10.6 本 章 小 结

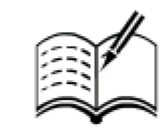
本章介绍的是用户界面设计中的高级部分，主要分为高级组件和对话框与消息提示两部分。在高级组件部分，主要介绍了两种日期时间类组件和 3 种进度条组件，其中，需要重点掌握的是 ProgressBar 进度条组件和 RatingBar 星级评分条组件的综合应用；在对话框与消息提示中，主要介绍了如何显示消息提示框、发送并显示通知，以及如何弹出各种对话框。在实际程序开发时，对话框和消息提示比较常用，需要读者重点掌握，并能做到融会贯通。

10.7 跟 我 上 机

参考答案：光盘\MR\跟我上机

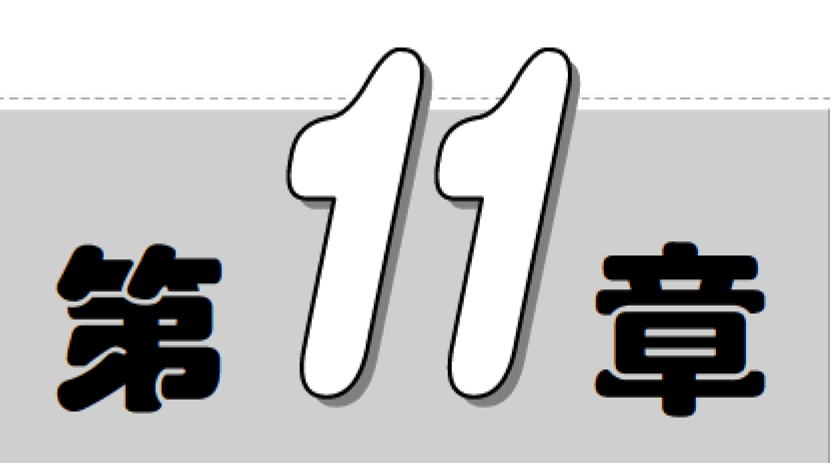
创建一个 Android 程序，实现弹出询问是否退出对话框的功能。具体实现时，首先需要在布局文件 main.xml 中将默认添加的 TextView 组件删除，并设置居中对齐，添加一个 ImageButton 组件，设置其背景透明；然后在主活动 MainActivity 的 onCreate()方法中，获取布局文件中添加的第一个按钮，也就是“退出”按钮，并为其添加单击事件监听器，在重写的 onClick()方法中，





使用 AlertDialog 类创建一个带“不”和“是的”按钮的提示对话框。创建对话框的参考代码如下：

```
AlertDialog alert = new AlertDialog.Builder(MainActivity.this)
        .create();
alert.setIcon(R.drawable.advise);                          //设置对话框的图标
alert.setTitle("退出？");                                  //设置对话框的标题
alert.setMessage("真的要退出泡泡龙游戏吗？");              //设置要显示的内容
//添加“取消”按钮
alert.setButton(DialogInterface.BUTTON_NEGATIVE, "不",
        new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        });
//添加“确定”按钮
alert.setButton(DialogInterface.BUTTON_POSITIVE, "是的",new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog,int which) {
                finish();                                  //返回系统主界面
            }
        });
alert.show();                                              //显示对话框
```



Android 中的事件处理

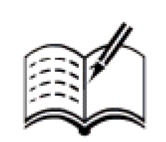
（视频讲解：20 分钟）

用户在使用手机、平板电脑时，总是通过各种操作来与软件进行交互，比较常见的方式包括键盘操作、触摸操作和手势等。在 Android 中，这些操作都转换为对应的事件进行处理，本章将对 Android 中的事件处理进行介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 屏蔽物理键盘中的后退键
- ☐ 显示短时间和长时间单击按钮信息
- ☐ 当用户触摸屏幕时显示提示信息
- ☐ 识别用户输入的手势
- ☐ 查看手势对应分值
- ☐ 使用手势输入数字
- ☐ 单击增加音量键时显示提示信息





11.1 事件处理概述

现代的图形界面应用程序，一般都是通过事件来实现人机交互的，事件就是用户对于图形界面的操作。在 Android 手机和平板电脑上，主要包括键盘事件和触摸事件两大类，其中，键盘事件包括按下、弹起等，而触摸事件包括按下、弹起、滑动和双击等。

在 Android 控件中，提供了事件处理的相关方法。例如，在 View 类中，提供了 onTouchEvent() 方法来处理触摸事件。但是，仅有重写这个方法才能完成事件处理显然并不实用，这种方式主要适用于重写控件的场景。除了 onTouchEvent()方法外，还可以使用 setOnTouchListener()方法为控件设置监听器来处理触摸事件，这在日常开发中更加常用。

11.2 处理键盘事件

对于一个标准的 Android 设备，包含了多个能够触发事件的物理按键，各个可用的物理按键能够触发的事件说明如表 11.1 所示。

表 11.1 Android 设备可用的物理按键

物理按键	KeyEvent	说明
电源键	KEYCODE_POWER	启动或唤醒设备，将界面切换到锁定的屏幕
后退键	KEYCODE_BACK	返回到前一个界面
菜单键	KEYCODE_MENU	显示当前应用的可用菜单
HOME 键	KEYCODE_HOME	返回到 HOME 界面
搜索键	KEYCODE_SEARCH	在当前应用中启动搜索
相机键	KEYCODE_CAMERA	启动相机
音量键	KEYCODE_VOLUME_UP KEYCODE_VOLUME_DOWN	控制当前上下文音量，如音乐播放器、手机铃声、通话音量等
方向键	KEYCODE_DPAD_CENTER KEYCODE_DPAD_UP KEYCODE_DPAD_DOWN KEYCODE_DPAD_LEFT KEYCODE_DPAD_RIGHT	某些设备中包含的方向键，用于移动光标等

Android 中的控件在处理物理按键事件时，提供的回调方法有 onKeyUp()、onKeyDown()和 onKeyLongPress()。

【例 11.1】 在 Eclipse 中创建 Android 项目，主要实现屏蔽物理键盘中的后退键的功能。

实例位置：光盘\MR\Instance\11\11.1

编写 ForbiddenBackActivity 类，重写 onCreate()方法来加载布局文件，重写 onKeyDown()方



法来拦截用户单击后退按钮事件。其代码如下：

```
public class ForbiddenBackActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                    //设置页面布局
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            return true;                                  //屏蔽后退键
        }
        return super.onKeyDown(keyCode, event);
    }
}
```

运行程序，显示如图 11.1 所示的界面，这时再单击后退键，可以看到应用程序并未退出。

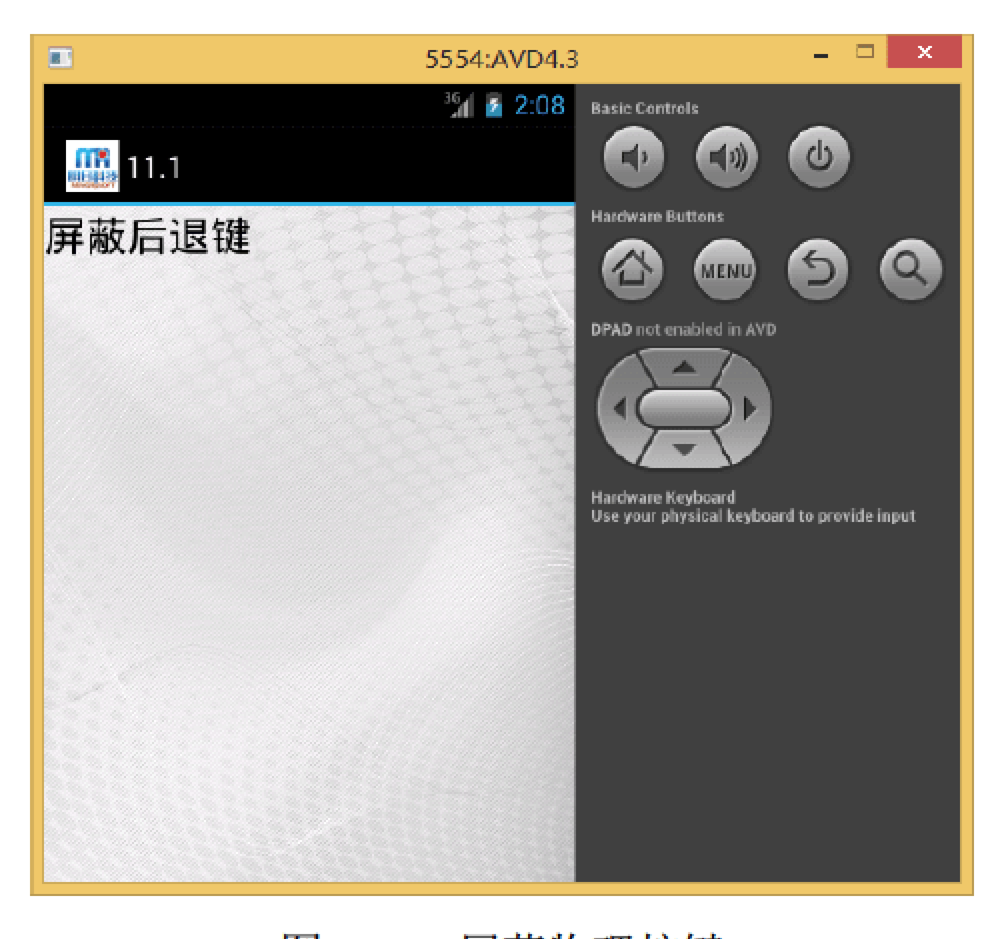


图 11.1　屏蔽物理按键

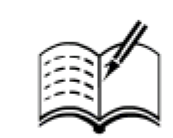
11.3　处理触摸事件

目前主流的手机都提供了大屏幕，从而取代了外置键盘，平板电脑也没有提供键盘，这些设备都需要通过触摸来操作。下面就介绍一下 Android 中如何实现触摸事件的处理。

对于触摸屏上的按钮，可以使用 OnClickListener 和 OnLongClickListener 两个监听器分别处理用户短时间单击和长时间单击（按住按钮一段时间）。下面通过一个实例来演示这两个方法的使用。

【例 11.2】 在 Eclipse 中创建 Android 项目，主要实现的功能是：当用户短时间单击按钮和长时间单击按钮时，显示不同的提示信息。





实例位置：光盘\MR\Instance\11\11.2

编写 ButtonTouchEventActivity 类，它继承自 Activity 类，重写 onCreate()方法来加载布局文件，使用 findViewById()方法获得布局文件中定义的按钮，为其增加了 OnClickListener 和 OnLongClickListener 两个事件监听器。其代码如下：

```
public class ButtonTouchEventActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                          //设置页面布局
        Button button = (Button) findViewById(R.id.button);     //获得按钮控件
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {                       //处理用户短时间单击按钮事件
                Toast.makeText(TouchEventActivity.this, getText(R.string.short_click),
                        Toast.LENGTH_SHORT).show();
            }
        });
        button.setOnLongClickListener(new OnLongClickListener() {
            public boolean onLongClick(View v) {                //处理用户长时间单击按钮事件
                Toast.makeText(TouchEventActivity.this, getText(R.string.long_click),
                        Toast.LENGTH_SHORT).show();
                return true;
            }
        });
    }
}
```

运行程序后，短时间单击按钮，显示如图 11.2 所示的提示信息。

长时间单击按钮，显示如图 11.3 所示的提示信息。

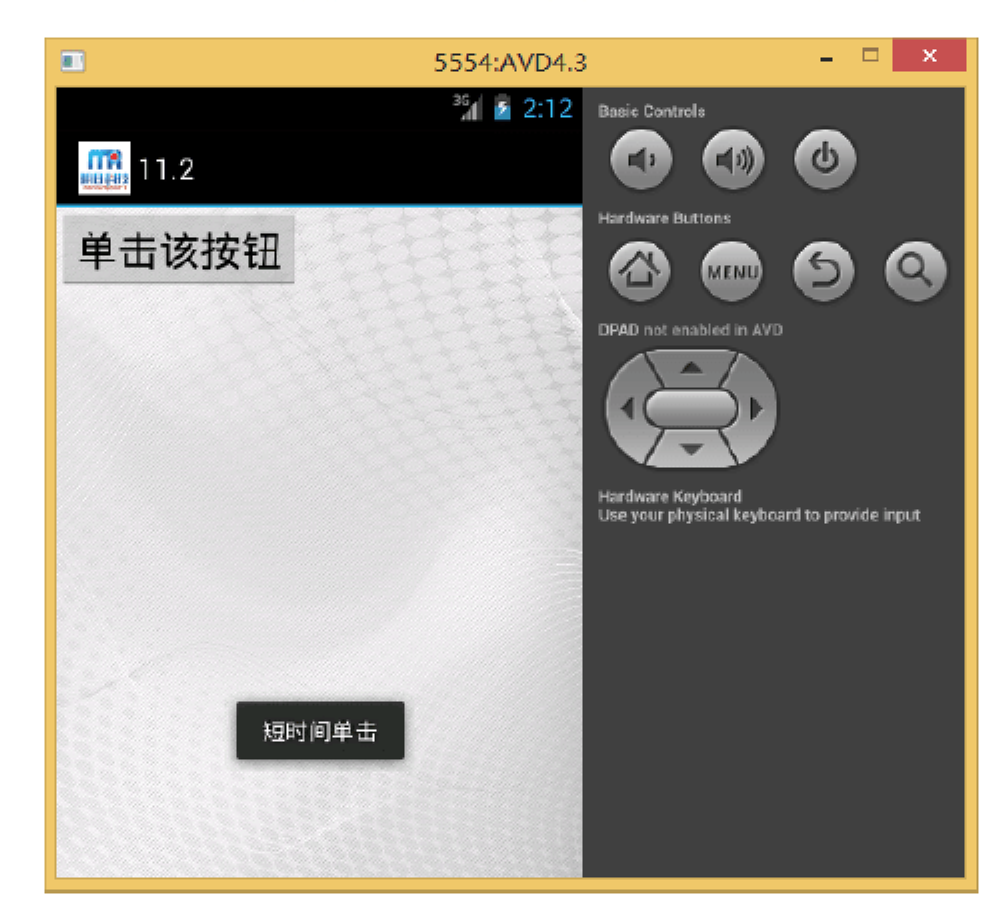



图 11.2　显示短时间单击按钮信息

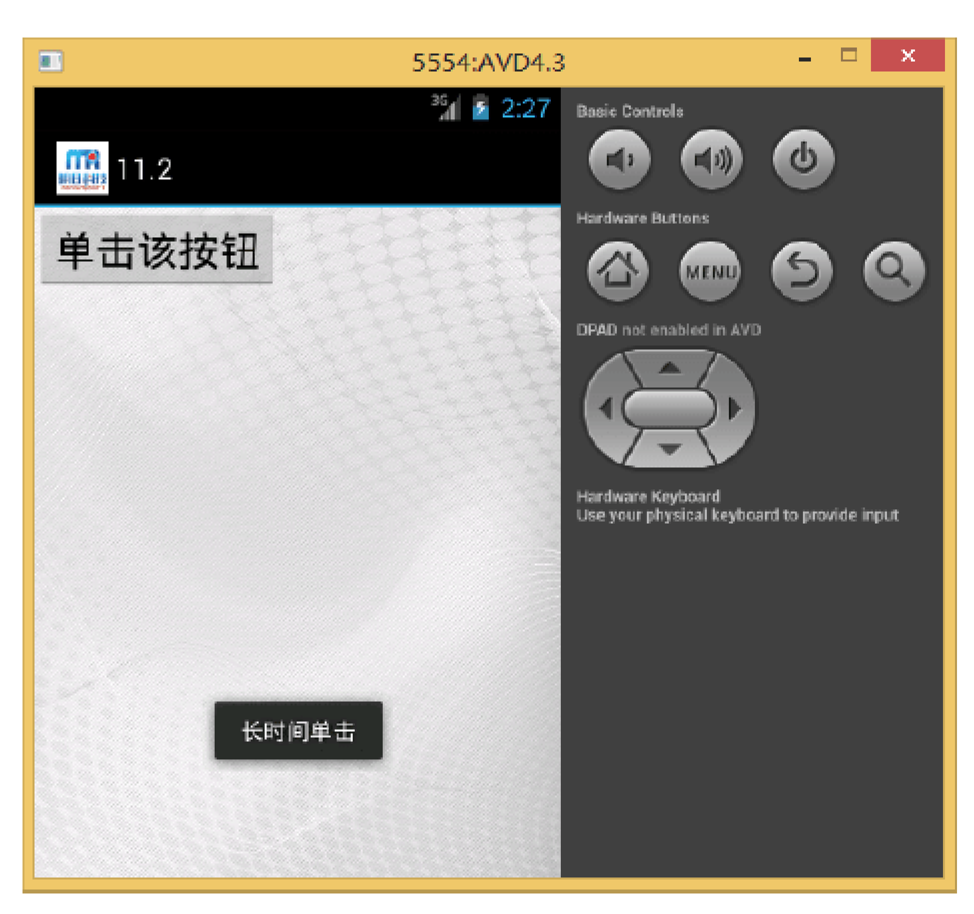


图 11.3　显示长时间单击按钮信息

View 类是其他 Android 控件的父类，在该类中，定义了 setOnTouchListener()方法，用来为



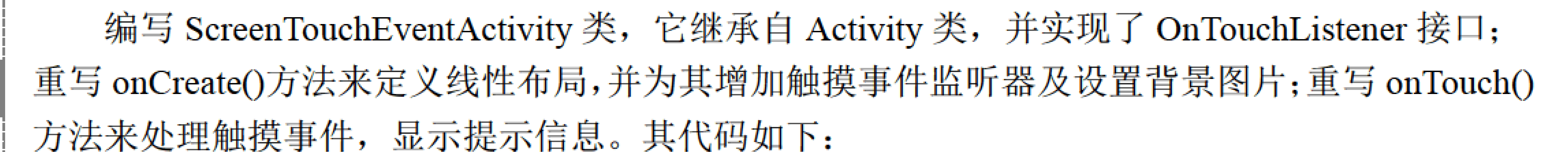


控件设置触摸事件监听器。下面演示该监听器的用法。

【例 11.3】 在 Eclipse 中创建 Android 项目，实现当用户触摸屏幕时显示提示信息的功能。



实例位置：光盘\MR\Instance\11\11.3



编写 ScreenTouchEventActivity 类，它继承自 Activity 类，并实现了 OnTouchListener 接口；重写 onCreate()方法来定义线性布局，并为其增加触摸事件监听器及设置背景图片；重写 onTouch()方法来处理触摸事件，显示提示信息。其代码如下：

```
public class ScreenTouchEventActivity extends Activity implements OnTouchListener {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                    //调用父类构造方法
        LinearLayout layout = new LinearLayout(this);          //定义线性布局
        layout.setOnTouchListener(this);                       //设置触摸事件监听器
        layout.setBackgroundResource(R.drawable.background);   //设置背景图片
        setContentView(layout);                                //使用布局
    }
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        Toast.makeText(this, "发生触摸事件", Toast.LENGTH_LONG).show();
        return true;
    }
}
```

运行程序后，触摸屏幕，显示如图 11.4 所示的提示信息。

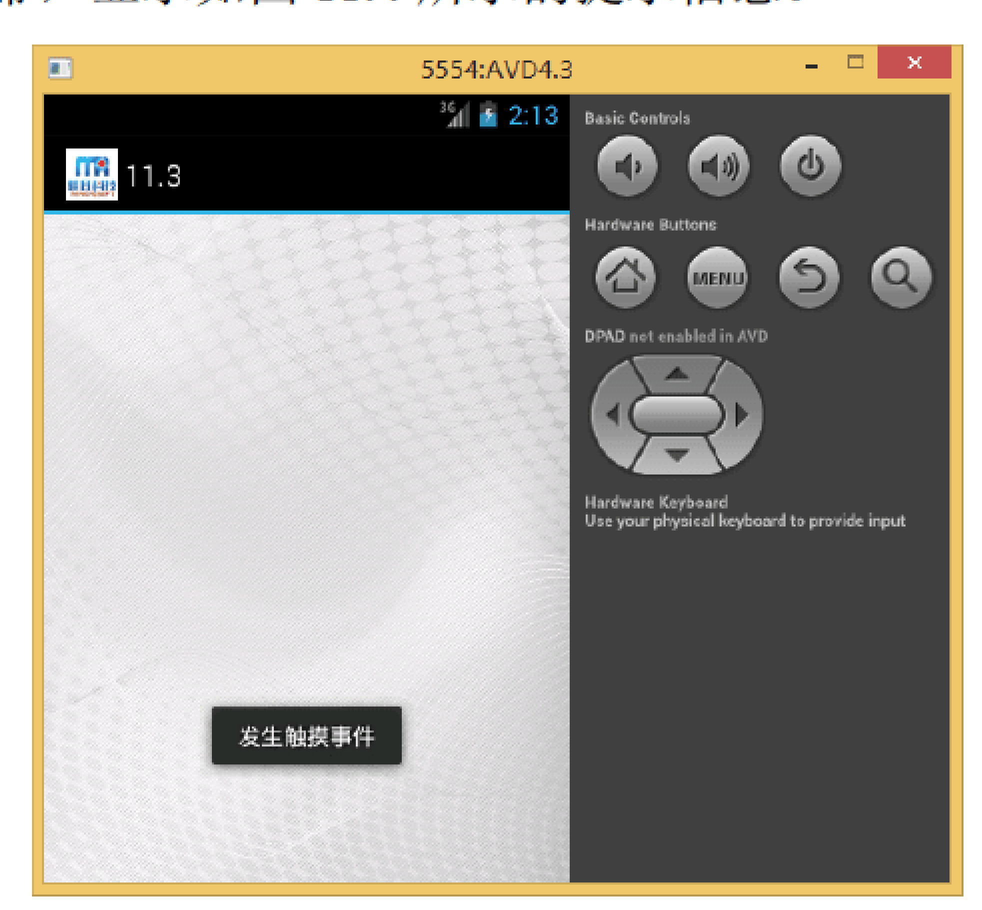
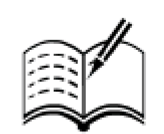


图 11.4　显示触摸事件信息

11.4　手势的创建与识别

前面介绍的触摸事件都比较简单，下面介绍一下手势在 Android 中如何创建和识别。目前，





大多数智能手机都支持手写输入，其原理就是根据用户输入的内容，在预先定义的词库中查找最佳的匹配项供用户选择。在 Android 中，也需要先定义类似的词库。

11.4.1　手势的创建

下面请读者运行自己的模拟器，进入到应用程序界面，如图 11.5 所示。

在图 11.5 中，单击 Gestures Builder 图标，进入如图 11.6 所示界面。



图 11.5　应用程序界面

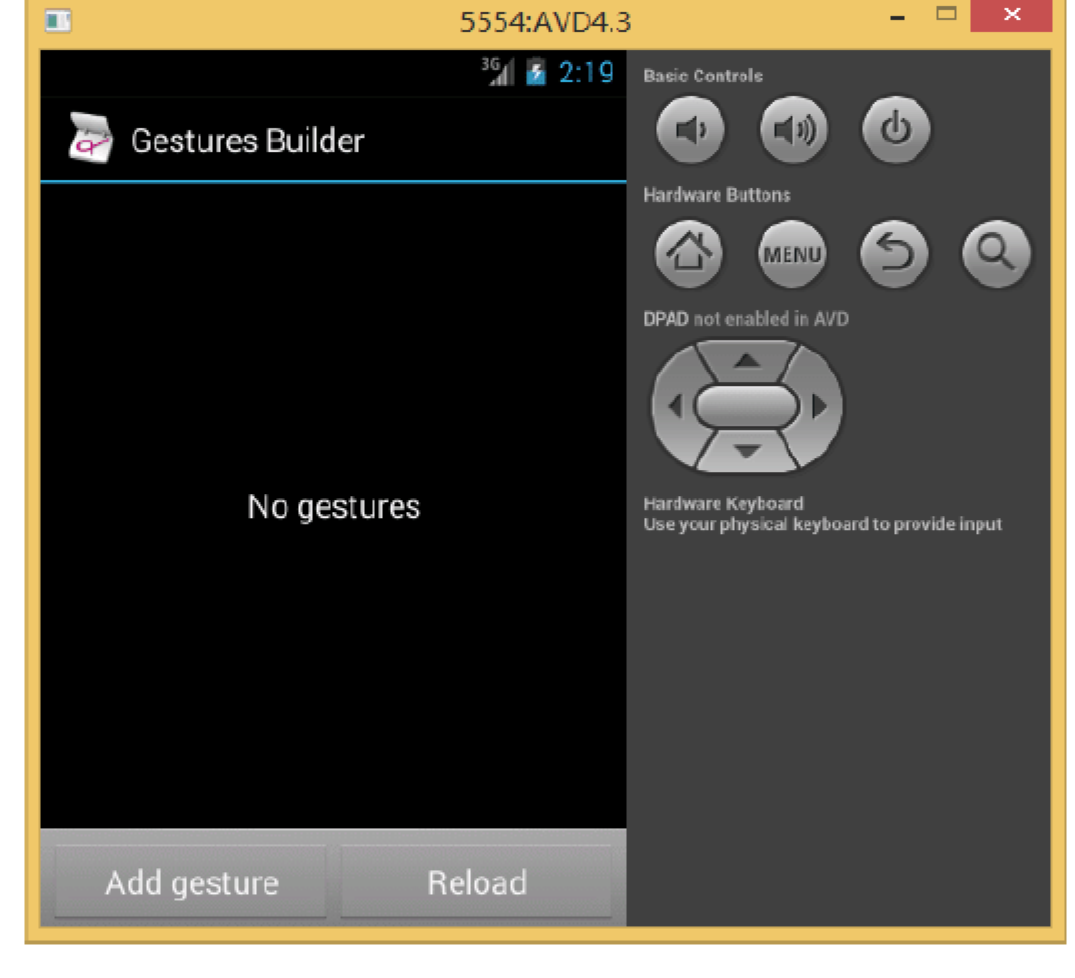


图 11.6　Gestures Builder 程序界面

在图 11.6 中，单击 Add gesture 按钮增加手势，如图 11.7 所示。在 Name 栏中输入该手势所代表的字符，在 Name 栏下方画出对应的手势，单击 Done 按钮完成手势的增加。

按照以上步骤继续增加数字 1、2、3……所对应的手势，如图 11.8 所示。

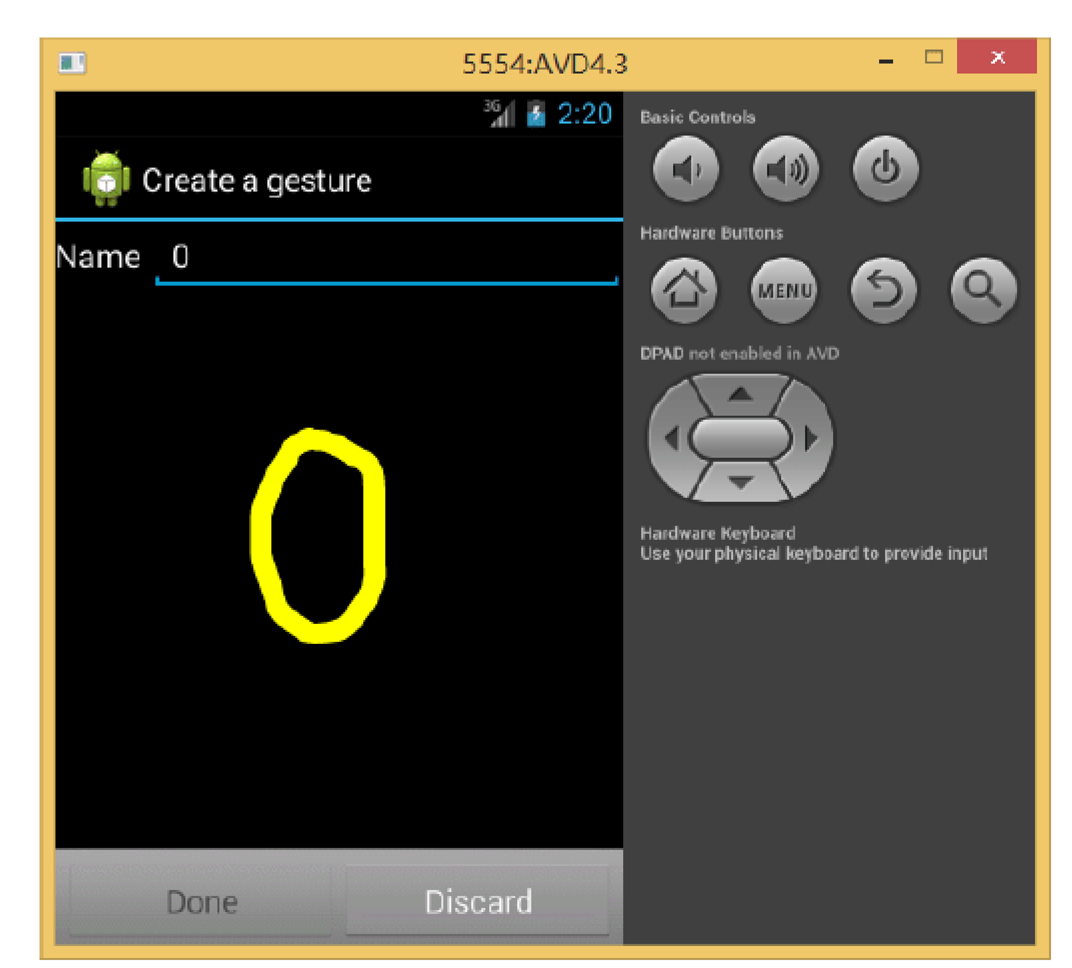


图 11.7　增加手势界面

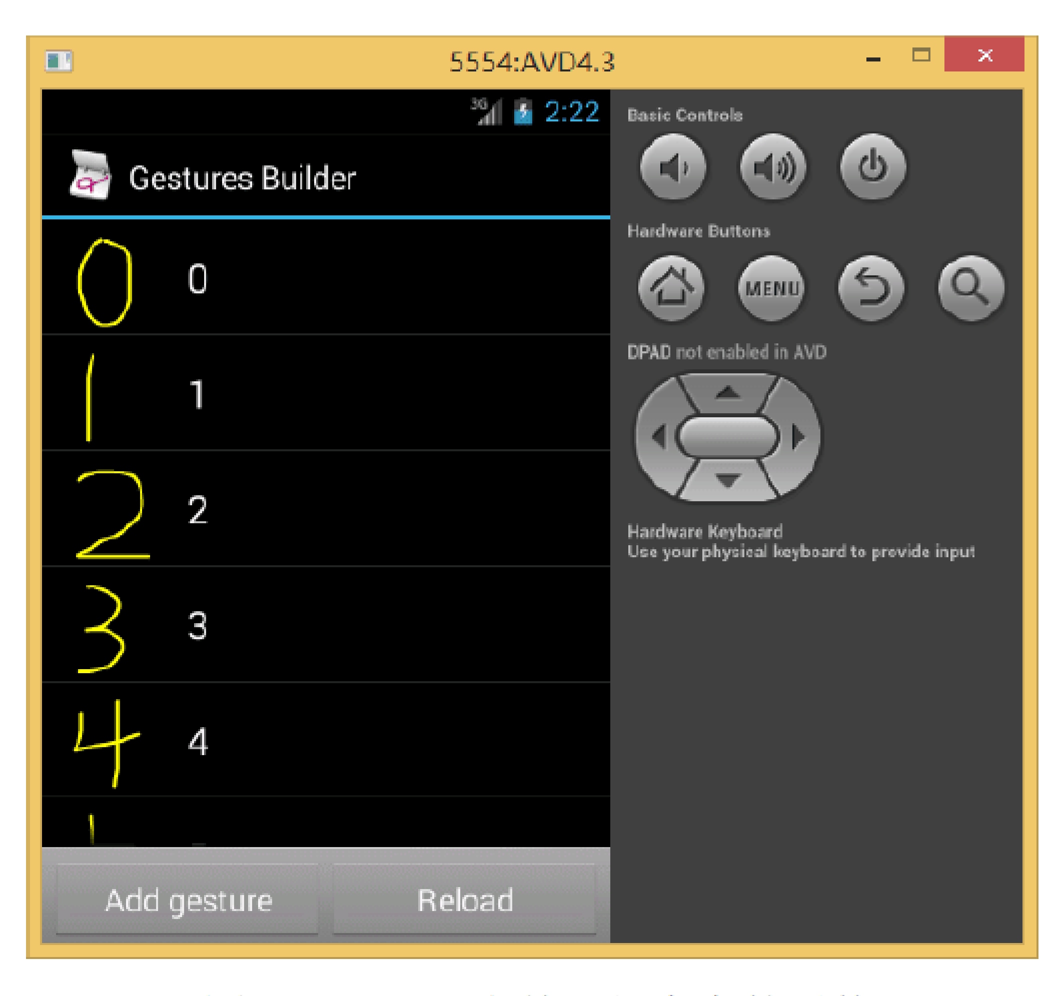
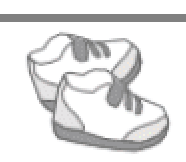
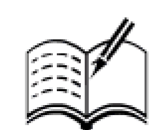


图 11.8　显示当前已经存在的手势

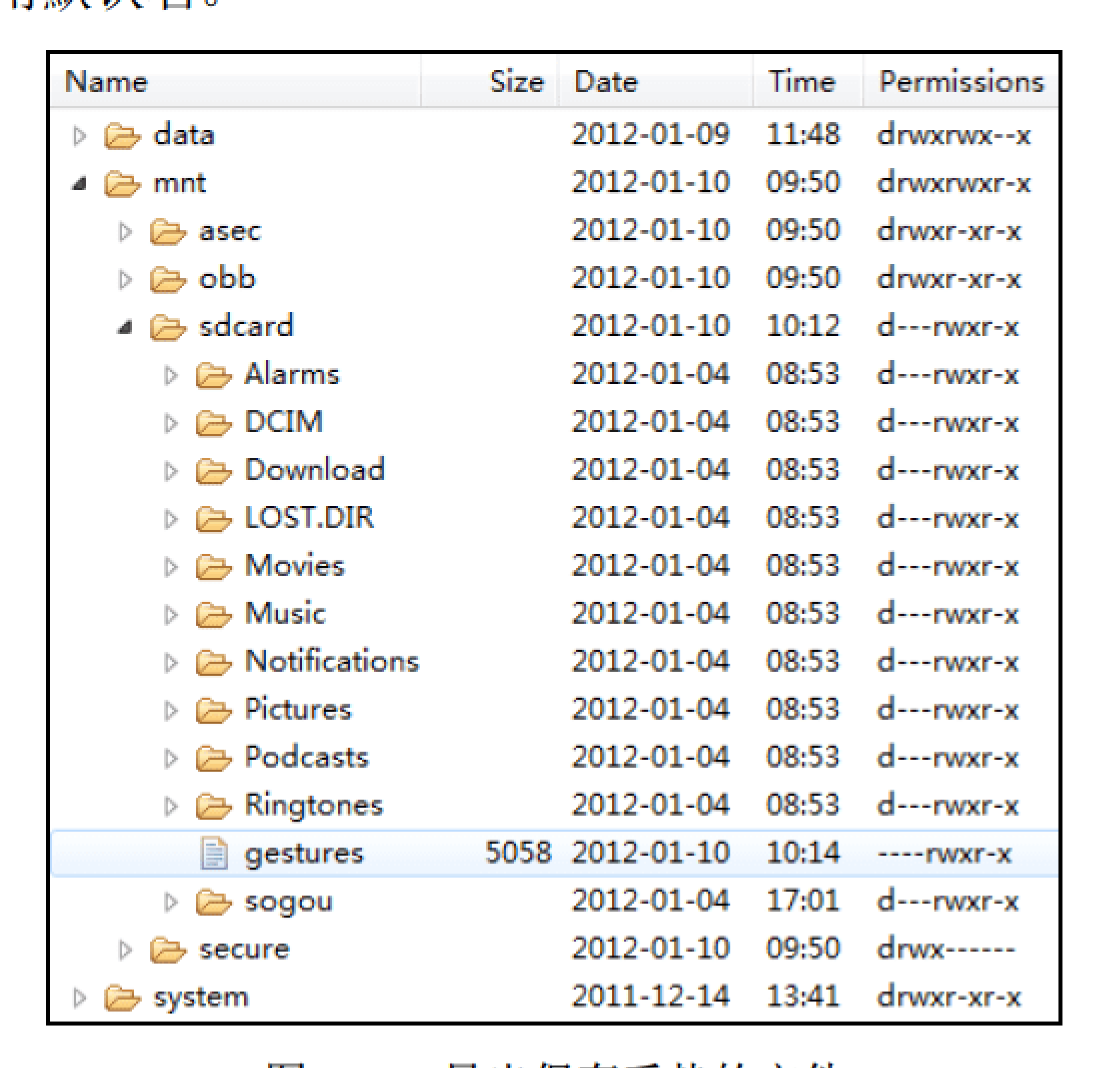




Note

11.4.2　手势的导出

在创建完手势后，需要将保存手势的文件导出，以便在开发的应用程序中使用。打开 Eclipse 并切换到 DDMS 视图。在 File Explorer 选项卡中，找到\mnt\sdcard\gestures 文件，如图 11.9 所示。将该文件导出，名称使用默认名。



Name	Size	Date	Time	Permissions
data		2012-01-09	11:48	drwxrwx--x
mnt		2012-01-10	09:50	drwxrwxr-x
asec		2012-01-10	09:50	drwxr-xr-x
obb		2012-01-10	09:50	drwxr-xr-x
sdcard		2012-01-10	10:12	d---rwxr-x
Alarms		2012-01-04	08:53	d---rwxr-x
DCIM		2012-01-04	08:53	d---rwxr-x
Download		2012-01-04	08:53	d---rwxr-x
LOST.DIR		2012-01-04	08:53	d---rwxr-x
Movies		2012-01-04	08:53	d---rwxr-x
Music		2012-01-04	08:53	d---rwxr-x
Notifications		2012-01-04	08:53	d---rwxr-x
Pictures		2012-01-04	08:53	d---rwxr-x
Podcasts		2012-01-04	08:53	d---rwxr-x
Ringtones		2012-01-04	08:53	d---rwxr-x
gestures	5058	2012-01-10	10:14	----rwxr-x
sogou		2012-01-04	17:01	d---rwxr-x
secure		2012-01-10	09:50	drwx------
system		2011-12-14	13:41	drwxr-xr-x

图 11.9　导出保存手势的文件

11.4.3　手势的识别

下面通过一个实例演示如何在 Android 程序中识别用户输入的手势。

【例 11.4】 在 Eclipse 中创建 Android 项目，实现识别用户输入手势的功能。

实例位置：光盘\MR\Instance\11\11.4

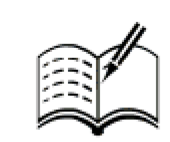
程序的开发步骤如下：

（1）在 res 文件夹中创建子文件夹，名称为 raw。将前面导出的手势文件复制到该文件夹中。

（2）在 main.xml 布局文件中，增加一个 GuestOverlayView 控件来接收用户的手势。main.xml 布局文件的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <TextView
        android:layout_width="fill_parent"
```






```
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:text="@string/title"
            android:textColor="@android:color/black"
            android:textSize="20dp" />
        <android.gesture.GestureOverlayView
            android:id="@+id/gestures"
            android:layout_width="fill_parent"
            android:layout_height="0dip"
            android:layout_weight="1.0" />
</LinearLayout>
```

（3）创建 GesturesRecognitionActivity 类，它继承自 Activity 类，并实现了 OnGesturePerformedListener 接口；在 onCreate()方法中，加载 raw 文件夹中的手势文件，并获得布局文件中定义的 GestureOverlayView 控件；在 onGesturePerformed()方法的实现中，获得得分最高的预测结果并提示。该类代码如下：

```
public class GesturesRecognitionActivity extends Activity implements OnGesturePerformedListener {
    private GestureLibrary library;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        library = GestureLibraries.fromRawResource(this, R.raw.gestures);    //加载手势文件
        if (!library.load()) {                                                 //如果加载失败则退出
            finish();
        }
        GestureOverlayView gesture = (GestureOverlayView) findViewById(R.id.gestures);
        gesture.addOnGesturePerformedListener(this);                           //增加事件监听器
    }
    @Override
    public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
        ArrayList<Prediction> gestures = library.recognize(gesture);           //获得全部预测结果
        int index = 0;                                                         //保存当前预测的索引号
        double score = 0.0;                                                    //保存当前预测的得分
        for (int i = 0; i < gestures.size(); i++) {                            //获得最佳匹配结果
            Prediction result = gestures.get(i);                               //获得一个预测结果
            if (result.score > score) {
                index = i;
                score = result.score;
            }
        }
        Toast.makeText(this, gestures.get(index).name, Toast.LENGTH_LONG).show();
    }
}
```

运行程序后，绘制手势，如图 11.10 所示。

在手势绘制完成后，显示提示信息，如图 11.11 所示。





Note





图 11.10　用户绘制的手势　　　　图 11.11　手势对应的信息

11.5 综合应用

11.5.1 查看手势对应分值

【例 11.5】 本实例主要根据用户绘制的手势显示其对应分值的功能，运行程序后，绘制手势，如图 11.12 所示。

实例位置：光盘\MR\Instance\11\11.5

在手势绘制完成后，根据绘制的手势显示其得分信息，如图 11.13 所示。



图 11.12　用户绘制的手势

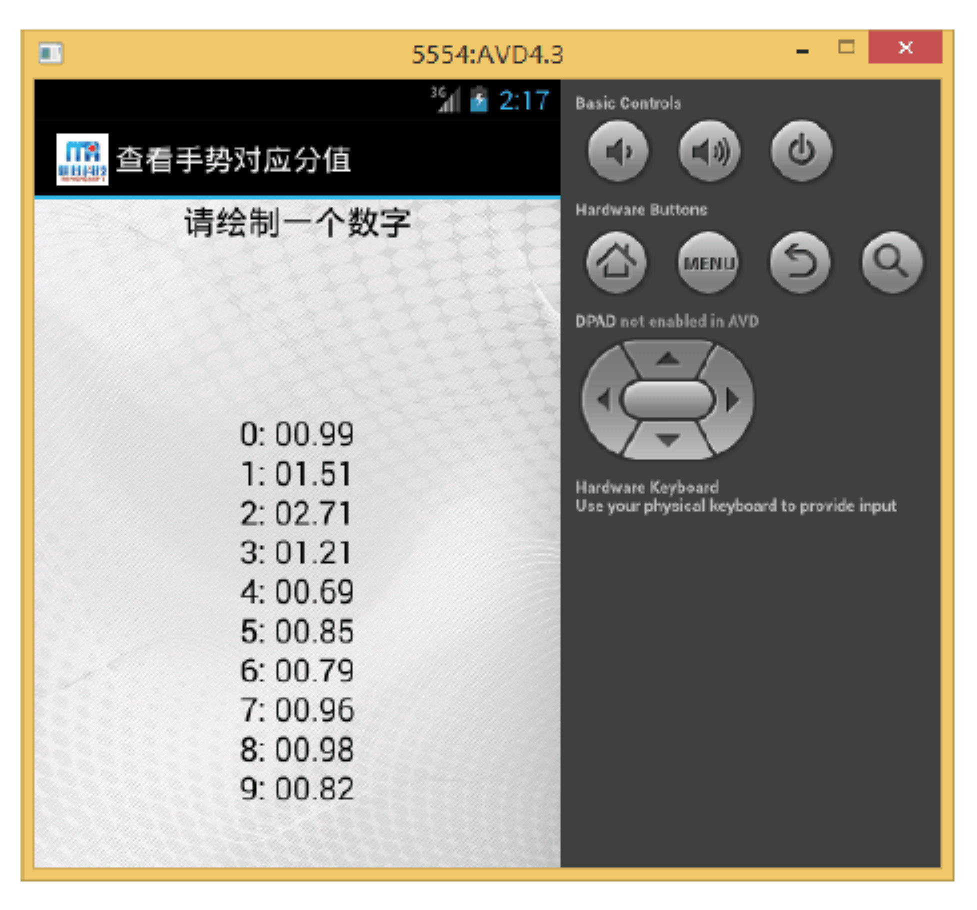
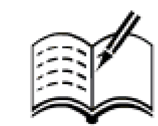


图 11.13　手势得到的分值

本实例实现时，首先需要实现 OnGesturePerformedListener 接口，并重写 Activity 的 onCreate()方法，在该方法中加载自定义的手势文件；然后获得布局文件中定义的 GestureOverlayView 控件，





并在 onGesturePerformed()重写方法中获得所有手势所对应的分值进行显示。其代码如下：

```
public class MainActivity extends Activity implements OnGesturePerformedListener {
    private GestureLibrary library;
    private TextView resultTV;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        library = GestureLibraries.fromRawResource(this, R.raw.gestures);    //加载手势文件
        resultTV = (TextView) findViewById(R.id.prediction);
        if (!library.load()) {                                               //如果加载失败则退出
            finish();
        }
        GestureOverlayView gesture = (GestureOverlayView) findViewById(R.id.gestures);
        gesture.addOnGesturePerformedListener(this);                         //增加事件监听器
    }
    @Override
    public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
        ArrayList<Prediction> gestures = library.recognize(gesture);         //获得全部预测结果
        Collections.sort(gestures, new Comparator<Prediction>() {            //将预测结果进行排序
                    @Override
                    public int compare(Prediction lhs, Prediction rhs) {
                        return lhs.name.compareTo(rhs.name);   //使用结果对应的字符串来排序
                    }
                });
        StringBuilder results = new StringBuilder();                         //保存全部结果
        NumberFormat formatter = new DecimalFormat("#00.00");                //定义格式化样式
        for (int i = 0; i < gestures.size(); i++) {                          //遍历全部结果
            Prediction result = gestures.get(i);
            results.append(result.name + ": " + formatter.format(result.score) + "\n");
        }
        resultTV.setText(results);                                           //显示结果
    }
}
```

11.5.2　使用手势输入数字

【例 11.6】 本实例主要实现利用用户绘制的手势在编辑框中输入数字的功能，运行程序后，绘制手势，如图 11.14 所示。

实例位置：光盘\MR\Instance\11\11.6

在手势绘制完成后，将与其最佳匹配的数字显示在编辑框中，如图 11.15 所示。





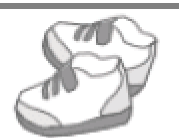

图 11.14　用户绘制的手势



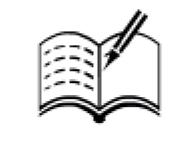
图 11.15　显示与手势最匹配的数字

本实例实现时，首先需要实现 OnGesturePerformedListener 接口，并重写 Activity 的 onCreate()方法，在该方法中加载自定义的手势文件；然后获得布局文件中定义的 GestureOverlayView 控件，并在 onGesturePerformed()重写方法中获得手势的最佳匹配数字进行显示。其代码如下：

```
public class NumberInputActivity extends Activity implements OnGesturePerformedListener {
    private GestureLibrary library;
    private EditText et;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        library = GestureLibraries.fromRawResource(this, R.raw.gestures);    //加载手势文件
        et = (EditText) findViewById(R.id.editText);
        if (!library.load()) {                                                //如果加载失败则退出
            finish();
        }
        GestureOverlayView gesture = (GestureOverlayView) findViewById(R.id.gestures);
        gesture.addOnGesturePerformedListener(this);                          //增加事件监听器
    }
    @Override
    public void onGesturePerformed(GestureOverlayView overlay, Gesture gesture) {
        ArrayList<Prediction> gestures = library.recognize(gesture); //获得全部预测结果
        int index = 0;                                                        //保存当前预测的索引号
        double score = 0.0;                                                   //保存当前预测的得分
        for (int i = 0; i < gestures.size(); i++) {                           //获得最佳匹配结果
            Prediction result = gestures.get(i);                              //获得一个预测结果
            if (result.score > score) {
                index = i;
                score = result.score;
            }
        }
        String text = et.getText().toString();                                //获得编辑框中已经包含的文本
        text += gestures.get(index).name;                                     //获得最佳匹配
```



```
            et.setText(text);                                              //更新编辑框
        }
    }
```



Note

11.6 本章常见错误

开发 Android 程序，想要通过手势左右滑动切换 Activity，但是其中一个 Activity 是使用 ListView 组件布局的，每次左右滑动时，都是在 ListView 组件的 Item 中滑动，如何解决这个问题呢？

该问题可以通过重写 ListView 组件所在 Activity 的 dispatchTouchEvent()方法来解决。例如，可以使用下面的代码。

```
@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
    mDetector.onTouchEvent(ev);
    return super.dispatchTouchEvent(ev);
}
```



说明：

上面代码中的 mDetector 是 GestrueDetector 的一个实例。

另外，除了上面的方法外，还可以考虑使用 ViewPager 组件来替换 ListView 组件，从而实现通过手势左右滑动切换 Activity 的功能。

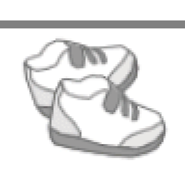
11.7 本 章 小 结

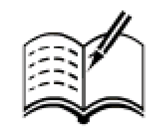
本章重点讲解了 Android 中常见的事件处理方式，通过与前面介绍的常用控件结合，可以实现 Android 应用程序的外部框架。本章的内容几乎在各种 Android 应用程序中都会用到，请读者务必熟练掌握。

11.8 跟 我 上 机

参考答案：光盘\MR\跟我上机

创建一个 Android 程序，主要实现当用户单击增加音量键时显示提示信息的功能。具体实现时，需要创建一个 VolumeUpMessageActivity 类，该类继承自 Activity 类。在 VolumeUpMessage




Note

Activity 类中重写 onCreate()方法来加载布局文件；重写 onKeyDown()方法，实现当音量增加键被按下时显示提示信息的功能。其代码如下：

```
public class VolumeUpMessageActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                              //设置页面布局
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
            Toast.makeText(this, "音量增加", Toast.LENGTH_LONG).show();   //提示音量增加
            return false;
        }
        return super.onKeyDown(keyCode, event);

    }

}
```

第12章

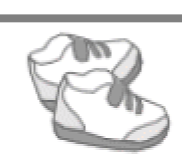
数据存储技术

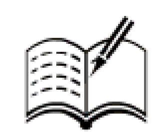
（视频讲解：44 分钟）

Android 为开发人员提供了多种持久化应用数据的方式，具体选择哪种方式需要具体问题具体分析，例如，数据是否仅限于本程序使用，还是可以用于其他程序，以及保存数据所占用的空间等。Android 中主要提供了 3 种数据存储技术，分别是 Shared Preferences、Files 和 SQLite 数据库，本章将对 Android 中的这 3 种数据存储技术进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 使用 SharedPreferences 保存用户输入的用户名和密码
- ☐ 使用 SharedPreferences 保存用户输入值
- ☐ 获取 SharedPreferences 中保存的值
- ☐ 使用内部存储保存用户输入的用户名和密码
- ☐ 在 SD 卡上创建文件
- ☐ 使用 SQLite 数据库保存用户输入的用户名和密码
- ☐ 遍历 Android 模拟器的 SD 卡
- ☐ 在 SQLite 数据库中批量添加数据
- ☐ 使用列表显示数据表中全部数据
- ☐ 复制图片到 SD 卡上





Note

12.1 使用 SharedPreferences 对象存储数据

SharedPreferences 类供开发人员保存和获取基本数据类型的键值对。该类主要用于基本类型，如 booleans、floats、ints、longs 和 strings。在应用程序结束后，数据仍旧会保存。

有两种方式可以获得 SharedPreferences 对象。

☑ getSharedPreferences()：如果需要多个使用名称来区分共享文件，则可以使用该方法，其第一个参数就是共享文件的名称。对于使用同一个名称获得的多个 SharedPreferences 引用，其指向同一个对象。

☑ getPreferences()：如果 Activity 仅需要一个共享文件，则可以使用该方法。因为只有一个文件，它并不需要提供名称。

完成向 SharedPreferences 类中增加值的步骤如下：

（1）调用 SharedPreferences 类的 edit()方法获得 SharedPreferences.Editor 对象。

（2）调用如 putBoolean()、putString()等方法增加值。

（3）使用 commit()方法提交新值。

从 SharedPreferences 类中读取值时，主要使用该类中定义的 getXxx()方法。下面以一个简单的实例演示 SharedPreferences 类的使用。

【例 12.1】 在 Eclipse 中创建 Android 项目，使用 SharedPreferences 保存用户输入的用户名和密码，并在第二个 Activity 中显示。

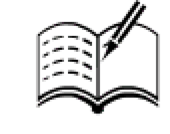
实例位置：光盘\MR\Instance\12\12.1

程序的开发步骤如下：

（1）修改\res\layout 包中的 main.xml 文件，增加文本框、编辑框等控件并修改它们的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/username"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
        <EditText
```



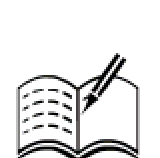


```
                android:id="@+id/username"
                android:layout_width="0dip"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:inputType="text"
                android:textColor="@android:color/white"
                android:textSize="20dp" >
                <requestFocus />
            </EditText>
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/password"
                android:textColor="@android:color/white"
                android:textSize="20dp" />
            <EditText
                android:id="@+id/password"
                android:layout_width="0dip"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:inputType="textPassword"
                android:textColor="@android:color/white"
                android:textSize="20dp" />
        </LinearLayout>
        <Button
            android:id="@+id/login"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/login"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
    </LinearLayout>
```

（2）创建 SharedPreferencesWriteActivity 类，重写 onCreate()方法，获得用户输入的用户名和密码，然后将其保存到 SharedPreferences 类中，最后使用 Intent 跳转到 SharedPreferences ReadActivity。其代码如下：

```
public class SharedPreferencesWriteActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                               //调用父类方法
        setContentView(R.layout.main);                                     //应用自定义布局文件
        final EditText usernameET = (EditText) findViewById(R.id.username); //用户名控件
        final EditText passwordET = (EditText) findViewById(R.id.password); //密码控件
        Button login = (Button) findViewById(R.id.login);                  //获得按钮控件
```





Note

```
            login.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String username = usernameET.getText().toString();        //获得用户名
                    String password = passwordET.getText().toString();        //获得密码
                    //获得私有类型的 SharedPreferences
                    SharedPreferences sp = getSharedPreferences("mrsoft", MODE_PRIVATE);
                    Editor editor = sp.edit();                                //获得 Editor 对象
                    editor.putString("username", username);                   //增加用户名
                    editor.putString("password", password);                   //增加密码
                    editor.commit();                                          //确认提交
                    Intent intent = new Intent();                             //创建 Intent 对象
                    //指定跳转到 SharedPreferencesReadActivity
                    intent.setClass(SharedPreferencesWriteActivity.this,
SharedPreferencesReadActivity.class);
                    startActivity(intent);                                    //实现跳转
                }
            });
        }
}
```

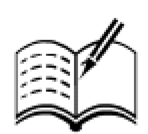
（3）在\res\layout 包中新建名为 result.xml 的布局文件，增加两个文本框并修改其默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
    <TextView
        android:id="@+id/password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
</LinearLayout>
```

（4）创建 SharedPreferencesReadActivity，它从 SharedPreferences 中读取已经保存的用户名和密码，然后使用文本框显示。其代码如下：

```
public class SharedPreferencesReadActivity extends Activity {
    @Override
```





```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                         //调用父类方法
        setContentView(R.layout.result);                            //设置布局文件
        TextView usernameTV = (TextView) findViewById(R.id.username);
        TextView passwordTV = (TextView) findViewById(R.id.password);
        //获得私有类型的 SharedPreferences
        SharedPreferences sp = getSharedPreferences("mrsoft", MODE_PRIVATE);
        String username = sp.getString("username", "mr");           //获得用户名
        String password = sp.getString("password", "001");          //获得密码
        usernameTV.setText("用户名：" + username);                   //显示用户名
        passwordTV.setText("密　　码：" + password);                 //显示密码
    }
}
```

（5）在 AndroidManifest.xml 文件中，定义两个 Activity 并配置启动项。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity android:name=".SharedPreferencesWriteActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SharedPreferencesReadActivity" />
    </application>
</manifest>
```

运行程序，显示如图 12.1 所示的用户登录界面。输入用户名“mr”和密码“123”，单击“登录”按钮，跳转到如图 12.2 所示的用户信息界面。



图 12.1　获得用户输入信息

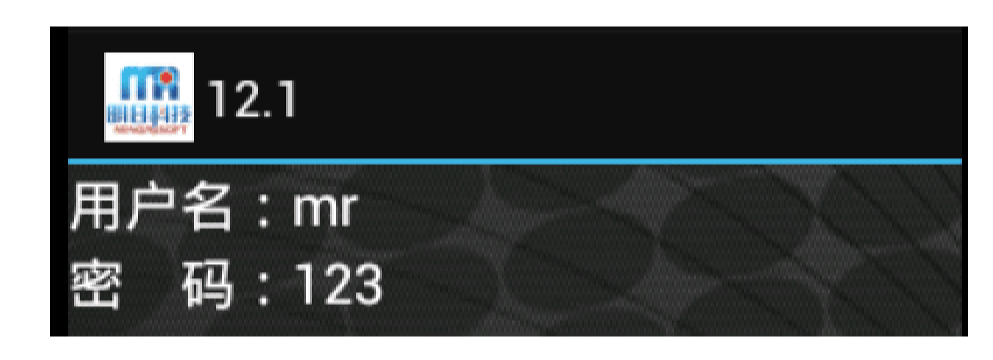
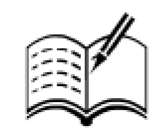


图 12.2　显示用户输入信息

对于 SharedPreferences 而言，它使用 XML 文件来保存数据，文件名与指定的名称相同。打开 DDMS 视图，在 File Explorer 中打开\data\data 文件夹可以看到如图 12.3 所示的文件。

在例 12.1 中，演示了如何使用私有的 SharedPreferences 来实现不同 Activity 之间的数据传递。除了 MODE_PRIVATE（默认模式）外，还有 MODE_WORLD_READABLE 和 MODE_WORLD_





Note

WRITEABLE 两种模式，它们分别表示对于其他应用程序而言，是否可读与可写。下面演示这两个模式的使用。

com.mingrisoft		2012-02-15	16:57	drwxr-x--x
cache		2012-02-15	16:52	drwxrwx--x
lib		2012-02-15	16:52	drwxr-xr-x
shared_prefs		2012-02-15	16:57	drwxrwx--x
mrsoft.xml	143	2012-02-15	16:57	-rw-rw----

图 12.3　XML 文件保存位置

【例 12.2】 在 Eclipse 中创建两个 Android 项目，分别命名为 1 和 2，在 1 中使用 SharedPreferences 保存用户输入值，在 2 中读取这些值。

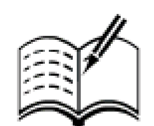
实例位置：光盘\MR\Instance\12\12.2

程序的开发步骤如下：

（1）在项目 1 中，修改\res\layout 包中的 main.xml 文件，增加文本框、编辑框、按钮等控件并修改它们的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/world_read"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
        <EditText
            android:id="@+id/worldRead"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="text"
            android:textColor="@android:color/white"
            android:textSize="20dp" >
            <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
```





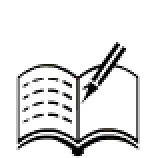
Note

```
            android:layout_height="wrap_content"
            android:text="@string/world_write"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
        <EditText
            android:id="@+id/worldWrite"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="text"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/word_read_write"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
        <EditText
            android:id="@+id/worldReadWrite"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:inputType="text"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
    </LinearLayout>
    <Button
        android:id="@+id/save"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
</LinearLayout>
```

（2）在项目 1 中，创建 SharedPreferencesWriteActivity 类，它位于 com.mingrisoft 包中，该类继承了 Activity 类。在该类中，创建了 3 个名称和权限都不相同的 SharedPreferences，向其中写入用户需要保存的值。其代码如下：

```
public class SharedPreferencesWriteActivity extends Activity {
    private EditText worldReadET;
    private EditText worldWriteET;
    private EditText worldReadWriteET;
    private SharedPreferences worldReadSP;
```





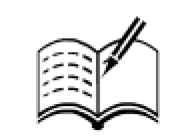
Note

```
    private SharedPreferences worldWriteSP;
    private SharedPreferences worldReadWriteSP;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                             //调用父类方法
        setContentView(R.layout.main);                                  //应用自定义布局文件
        worldReadET = (EditText) findViewById(R.id.worldRead);          //获得全局可读控件
        worldWriteET = (EditText) findViewById(R.id.worldWrite);        //获得全局可写控件
        worldReadWriteET = (EditText) findViewById(R.id.worldReadWrite);    //获得全局可读可写控件
        worldReadSP = getSharedPreferences("worldRead", MODE_WORLD_READABLE);
        worldWriteSP = getSharedPreferences("worldWrite", MODE_WORLD_WRITEABLE);
        worldReadWriteSP = getSharedPreferences("worldReadWrite", MODE_WORLD_READABLE +
MODE_WORLD_WRITEABLE);
        Button save = (Button) findViewById(R.id.save);
        save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String worldReadS = worldReadET.getText().toString();
                String worldWriteS = worldWriteET.getText().toString();
                String worldReadWriteS = worldReadWriteET.getText().toString();
                Editor worldReadE = worldReadSP.edit();
                Editor worldWriteE = worldWriteSP.edit();
                Editor worldReadWriteE = worldReadWriteSP.edit();
                worldReadE.putString("key", worldReadS);
                worldWriteE.putString("key", worldWriteS);
                worldReadWriteE.putString("key", worldReadWriteS);
                worldReadE.commit();
                worldWriteE.commit();
                worldReadWriteE.commit();
            }
        });
    }
}
```

（3）在项目 2 中，修改\res\layout 包中的 main.xml 文件，增加文本框控件并修改它们的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/worldRead"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
    <TextView
```






```
            android:id="@+id/worldWrite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
    <TextView
            android:id="@+id/worldReadWrite"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@android:color/white"
            android:textSize="20dp" />
</LinearLayout>
```

（4）在项目 2 中，创建 SharedPreferencesReadActivity 类，它位于 com.mingrisoft.other 包中，该类继承了 Activity 类。在该类中，获得在项目 1 中定义的 SharedPreference，然后显示其值。其代码如下：

```
public class SharedPreferencesReadActivity extends Activity {
    private SharedPreferences worldReadSP;
    private SharedPreferences worldWriteSP;
    private SharedPreferences worldReadWriteSP;
    private TextView worldReadTV;
    private TextView worldWriteTV;
    private TextView worldReadWriteTV;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Context otherContext = null;
        try {
            otherContext = createPackageContext("com.mingrisoft", MODE_PRIVATE);
        } catch (NameNotFoundException e) {
            e.printStackTrace();
        }
        worldReadSP = otherContext.getSharedPreferences("worldRead", MODE_WORLD_READABLE);
        worldWriteSP = otherContext.getSharedPreferences("worldWrite", MODE_WORLD_WRITEABLE);
        worldReadWriteSP = otherContext.getSharedPreferences("worldReadWrite", MODE_WORLD_
READABLE + MODE_WORLD_WRITEABLE);
        worldReadTV = (TextView) findViewById(R.id.worldRead);
        worldWriteTV = (TextView) findViewById(R.id.worldWrite);
        worldReadWriteTV = (TextView) findViewById(R.id.worldReadWrite);
        worldReadTV.setText("全局可读：" + worldReadSP.getString("key", "null"));
        worldWriteTV.setText("全局可写：" + worldWriteSP.getString("key", "null"));
        worldReadWriteTV.setText("全局可读可写：" + worldReadWriteSP.getString("key", "null"));
    }
}
```

运行项目 1，显示如图 12.4 所示的接收用户信息界面，全部输入“mr”，单击“保存键值对”按钮。



运行项目 2，显示如图 12.5 所示的界面。界面上显示了用户刚刚输入信息的获取情况。



图 12.4　接收用户信息界面

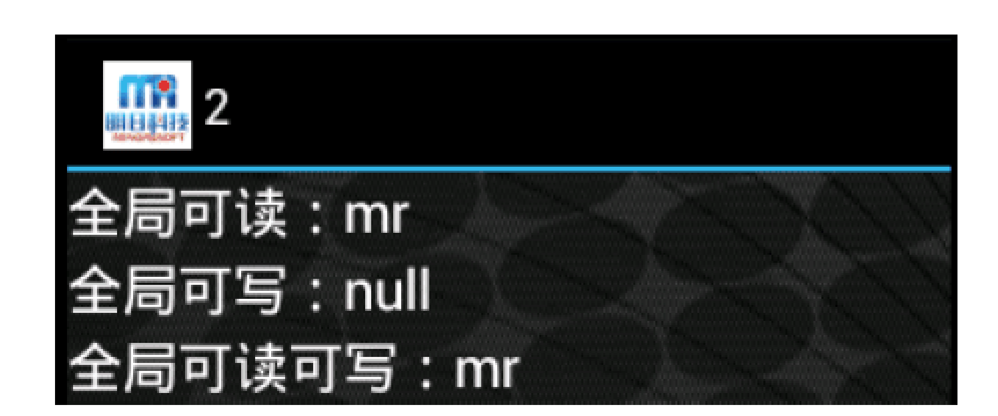


图 12.5　显示获得的信息

12.2　使用 Files 对象存储数据

在 Android 中，使用 Files 对象存储数据主要有两种方式：一种是 Java 提供的 IO 流体系，即使用 FileOutputStream 类提供的 openFileOutput()方法和 FileInputStream 类提供的 openFileInput()方法访问磁盘上的内容文件；另一种是使用 Environment 类的 getExternalStorageDirectory()方法对 Android 模拟器的 SD 卡进行数据读写。本节将对这两种方式进行详细讲解。

12.2.1　openFileOutput()和 openFileInput()方法

使用 Java 提供的 IO 流体系可以很方便地对 Android 模拟器本地存储的数据进行读写操作，其中，FileOutputStream 类的 openFileOutput()方法用来打开相应的输出流；而 FileInputStream 类的 openFileInput()方法用来打开相应的输入流。默认情况下，使用 IO 流保存的文件仅对当前应用程序可见，对于其他应用程序（包括用户）是不可见的（即不能访问其中的数据）。如果用户卸载了该应用程序，则保存数据的文件也会一起被删除。

下面通过一个实例演示如何使用 Java 提供的 IO 流体系对 Android 程序中的本地文件进行操作。

【例 12.3】 在 Eclipse 中创建 Android 项目，使用内部存储保存用户输入的用户名和密码，并在第二个 Activity 中显示。

实例位置：光盘\MR\Instance\12\12.3

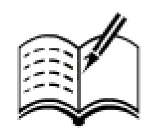
程序的开发步骤如下：

（1）本实例使用的布局文件与例 12.1 相同，请读者参考前面给出的代码。

（2）创建 InternalDataWriteActivity 类，重写 onCreate()方法，获得用户输入的用户名和密码，然后将其保存到 login 文件中，最后使用 Intent 跳转到 InternalDataReadActivity。其代码如下：

```
public class InternalDataWriteActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
```



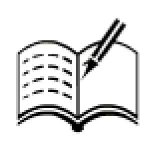


```
        super.onCreate(savedInstanceState);                                    //调用父类方法
        setContentView(R.layout.main);                                         //应用布局文件
        final EditText usernameET = (EditText) findViewById(R.id.username);    //用户名控件
        final EditText passwordET = (EditText) findViewById(R.id.password);    //密码控件
        Button login = (Button) findViewById(R.id.login);                      //获得按钮控件
        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String username = usernameET.getText().toString();             //获得用户名
                String password = passwordET.getText().toString();             //获得密码
                FileOutputStream fos = null;
                try {
                    fos = openFileOutput("login", MODE_PRIVATE);               //获得文件输出流
                    fos.write((username + " " + password).getBytes());         //保存用户名和密码
                    fos.flush();                                               //清除缓存
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                } finally {
                    if (fos != null) {
                        try {
                            fos.close();                                       //关闭文件输出流
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }
                Intent intent = new Intent();                                  //创建 Intent 对象
                //指定跳转到 InternalDataReadActivity
                intent.setClass(InternalDataWriteActivity.this, InternalDataReadActivity.class);
                startActivity(intent);                                         //实现跳转
            }
        });
    }
}
```

（3）创建 InternalDataReadActivity，它从 login 文件中读取已经保存的用户名和密码，然后使用文本框显示。其代码如下：

```
public class InternalDataReadActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                                    //调用父类方法
        setContentView(R.layout.result);                                       //使用布局文件
        FileInputStream fis = null;
        byte[] buffer = null;
        try {
            fis = openFileInput("login");                                      //获得文件输入流
            buffer = new byte[fis.available()];                                //定义保存数据的数组
            fis.read(buffer);                                                  //从输入流中读取数据
```





Note

```
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                if (fis != null) {
                    try {
                        fis.close();                                          //关闭文件输入流
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            TextView usernameTV = (TextView) findViewById(R.id.username);
            TextView passwordTV = (TextView) findViewById(R.id.password);
            String data = new String(buffer);                                 //获得数组中保存的数据
            String username = data.split(" ")[0];                             //获得 username
            String password = data.split(" ")[1];                             //获得 password
            usernameTV.setText("用户名： " + username);                        //显示用户名
            passwordTV.setText("密      码： " + password);                    //显示密码
        }
}
```

（4）在 AndroidManifest.xml 文件中定义两个 Activity 并配置启动项。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".InternalDataWriteActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".InternalDataReadActivity" />
    </application>
</manifest>
```

运行程序，显示如图 12.6 所示的用户登录界面。输入用户名“mr”和密码“123”，单击“登录”按钮，跳转到如图 12.7 所示的用户信息界面。





图 12.6　获得用户输入信息

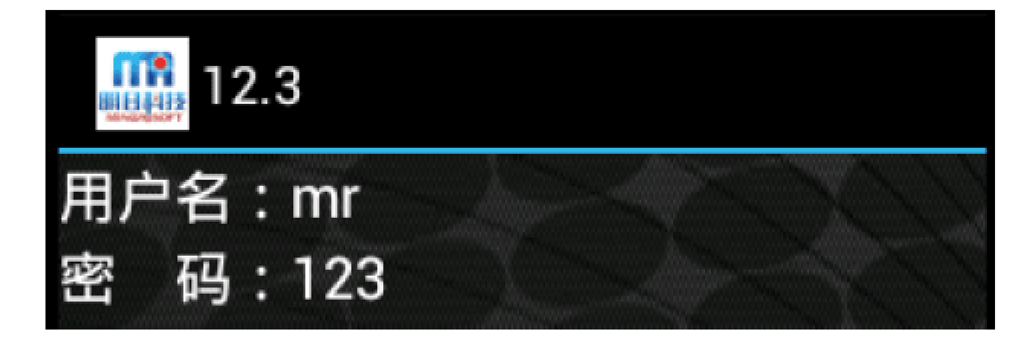


图 12.7　显示用户输入信息



Note

将 Eclipse 切换到 DDMS 视图，打开 File Explorer 中的 data\data 文件夹，可以看到保存数据的文件位于如图 12.8 所示的位置。

com.mingrisoft		2012-02-15	17:57	drwxr-x--x
cache		2012-02-15	16:52	drwxrwx--x
files		2012-02-15	17:57	drwxrwx--x
login	6	2012-02-15	18:04	-rw-rw----
lib		2012-02-15	16:52	drwxr-xr-x
shared_prefs		2012-02-15	16:57	drwxrwx--x

图 12.8　login 文件保存位置

12.2.2　对 Android 模拟器中的 SD 卡进行操作

每个 Android 设备都支持共享的外部存储用来保存文件，这可以是 SD 卡等可以移除的存储介质，也可以是手机内存等不可以移除的存储介质。保存的外部存储的文件都是全局可读的，而且在用户使用 USB 连接计算机后，可以修改这些文件。在 Android 程序中，对 SD 卡等外部存储的文件进行操作时，需要使用 Environment 类的 getExternalStorageDirectory()方法，该方法用来获取外部存储器（SD 卡）的目录。

【例 12.4】　在 Eclipse 中创建 Android 项目，实现在 SD 卡上创建文件的功能。

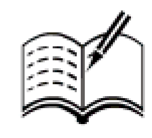
实例位置：光盘\MR\Instance\12\12.4

程序的开发步骤如下：

（1）修改 res\layout 包中的 main.xml 文件，在该文件中定义一个文本框并修改它的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="20dp" />
</LinearLayout>
```

（2）创建 FileCreateActivity 类，重写 onCreate()方法，使用 getExternalStorageDirectory()方法获得 SD 卡根文件夹，然后使用 createNewFile()方法创建文件并给出提示。其代码如下：


Note

```
public class FileCreateActivityextends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                    //调用父类方法
        setContentView(R.layout.main);                         //应用布局文件
        TextView tv = (TextView) findViewById(R.id.message);
        File root = Environment.getExternalStorageDirectory(); //获得 SD 卡根路径
        if(root.exists()&&root.canWrite()){
            File file = new File(root, "DemoFile.png");
            try {
                if (file.createNewFile()) {
                    tv.setText(file.getName() + "创建成功！");
                } else {
                    tv.setText(file.getName() + "创建失败！");
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }else {
            tv.setText("SD 卡不存在或者不可写！");
        }
    }
}
```

（3）修改 AndroidManifest.xml 配置文件，增加外部存储写入权限。修改完成后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="15" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".FileCreateActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

运行程序，显示如图 12.9 所示的文件创建成功信息。





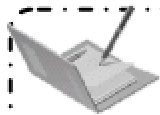
12.4
DemoFile.png创建成功！

图 12.9　文件创建成功

12.3　SQLite 数据库编程

对于更加复杂的数据结构，Android 提供了内置的 SQLite 数据库来存储数据。SQLite 使用 SQL 命令提供了完整的关系型数据库能力。每个使用 SQLite 的应用程序都有一个该数据库的实例，并且在默认情况下仅限当前应用使用。数据库存储在 Android 设置的\data\data\<package_name>\databases 文件夹中。使用 SQLite 数据库的步骤如下：

（1）创建数据库。
（2）打开数据库。
（3）创建表。
（4）完成数据的增删改查操作。
（5）关闭数据库。



说明：
关于 SQLite 支持的数据类型等信息请参考其官方文档。

【例 12.5】 在 Eclipse 中创建 Android 项目，使用 SQLite 数据库保存用户输入的用户名和密码，并在第二个 Activity 中显示。

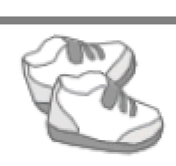
实例位置：光盘\MR\Instance\12\12.5

程序的开发步骤如下：

（1）本实例使用的布局文件与例 12.1 相同，请读者参考前面给出的代码。

（2）在 com.mingrisoft.util 包中创建 User 类，用来封装用户写入的信息。其代码如下：

```
public class User {
    private int id;                                   //保存用户的 id
    private String username;                          //保存用户名
    private String password;                          //保存密码
    public User() {
    }
    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }
    public int getId() {
        return id;
    }
    public String getUsername() {
        return username;
```



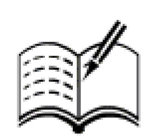
Note

```
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

（3）在 com.mingrisoft.util 包中创建 DBHelper 类，其中定义了若干字段来保存与数据库相关的信息。DBOpenHelper 类继承了 SQLiteOpenHelper 类，它提供了创建表格的功能。insert()方法用于向数据库表格中保存数据，query()方法用于根据 id 值来查询数据。其代码如下：

```
public class DBHelper {
    private static final String DATABASE_NAME = "datastorage";    //保存数据库名称
    private static final int DATABASE_VERSION = 1;                //保存数据库版本号
    private static final String TABLE_NAME = "users";             //保存表名称
    private static final String ID = "_id";                       //保存 id 值
    private static final String USERNAME = "username";            //保存用户名
    private static final String PASSWORD = "password";            //保存密码
    private DBOpenHelper helper;
    private SQLiteDatabase db;
    private static class DBOpenHelper extends SQLiteOpenHelper {
        //定义创建表格的 SQL 语句
        private static final String CREATE_TABLE = "create table " + TABLE_NAME + " ( " + ID + "
integer primary key autoincrement, " + USERNAME + " text not null, " + PASSWORD + " text not null);";
        public DBOpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            db.execSQL(CREATE_TABLE);                             //创建表格
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            db.execSQL("drop table if exists " + TABLE_NAME);     //删除旧版表格
            onCreate(db);                                         //创建表格
        }
    }
    public DBHelper(Context context) {
        helper = new DBOpenHelper(context);                   //创建 SQLiteOpenHelper 对象
        db = helper.getWritableDatabase();                    //获得可写的数据库
    }
    public void insert(User user) {                           //向表格中插入数据
        ContentValues values = new ContentValues();
        values.put(USERNAME, user.getUsername());
```





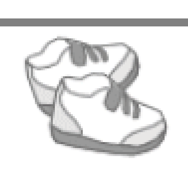

```
            values.put(PASSWORD, user.getPassword());
            db.insert(TABLE_NAME, null, values);
        }
        public User query(int id) {                                    //根据 id 值查询数据
            User user = new User();
            Cursor cursor = db.query(TABLE_NAME, new String[] { USERNAME, PASSWORD }, "_id = " +
id, null, null, null, null);
            if (cursor.getCount() > 0) {                               //如果获得的查询记录条数大于 0
                cursor.moveToFirst();                                  //将游标移动到第一条记录
                user.setUsername(cursor.getString(0));                 //获得用户名的值然后进行设置
                user.setPassword(cursor.getString(1));                 //获得密码的值然后进行设置
                return user;
            }
            cursor.close();                                            //关闭游标
            return null;
        }
}
```

（4）创建 SQLiteWriteActivity 类，重写 onCreate()方法，获得用户输入的用户名和密码，然后将其保存到 SQLite 数据库中，最后使用 Intent 跳转到 SQLiteReadActivity。其代码如下：

```
public class SQLiteWriteActivity extends Activity {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);                                    //调用父类方法
            setContentView(R.layout.main);                                         //应用自定义布局文件
            final EditText usernameET = (EditText) findViewById(R.id.username); //用户名控件
            final EditText passwordET = (EditText) findViewById(R.id.password);  //密码控件
            Button login = (Button) findViewById(R.id.login);                      //获得按钮控件
            login.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    String username = usernameET.getText().toString();     //获得用户名
                    String password = passwordET.getText().toString();     //获得密码
                    User user = new User(username, password);
                    DBHelper helper = new DBHelper(SQLiteWriteActivity.this);
                    helper.insert(user);                                   //向表格中插入数据
                    Intent intent = new Intent();                          //创建 Intent 对象
                    //指定跳转到 SQLiteReadActivity
                    intent.setClass(SQLiteWriteActivity.this, SQLiteReadActivity.class);
                    startActivity(intent);                                 //实现跳转
                }
            });
        }
}
```

（5）创建 SQLiteReadActivity 类，它从 SQLite 数据库中读取已经保存的用户名和密码，然后使用文本框显示。其代码如下：

```
public class SQLiteReadActivity extends Activity {
        @Override
```



```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                          //调用父类方法
        setContentView(R.layout.result);                             //设置布局文件
        TextView usernameTV = (TextView) findViewById(R.id.username);
        TextView passwordTV = (TextView) findViewById(R.id.password);
        DBHelper helper = new DBHelper(SQLiteReadActivity.this);
        User user = helper.query(1);
        usernameTV.setText("用户名：" + user.getUsername());           //显示用户名
        passwordTV.setText("密    码：" + user.getPassword());         //显示密码
    }
}
```

运行程序，显示如图 12.10 所示的用户登录界面。输入用户名“mr”和密码“123”，单击“登录”按钮，跳转到如图 12.11 所示的用户信息界面。



图 12.10　获得用户输入信息

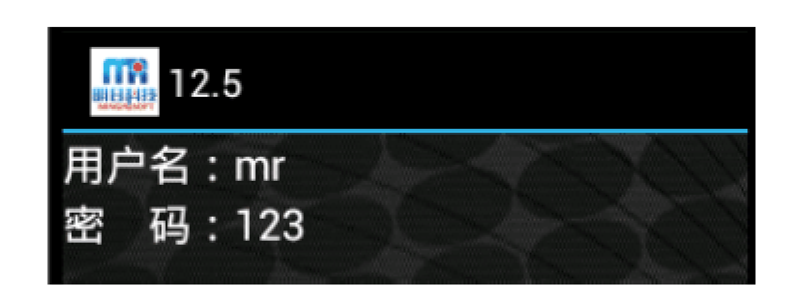


图 12.11　显示用户输入信息

打开 Eclipse 的 DDMS 视图，在 File Explorer 中打开\data\data 文件夹，可以看到 SQLite 数据库文件保存在如图 12.12 所示的位置。

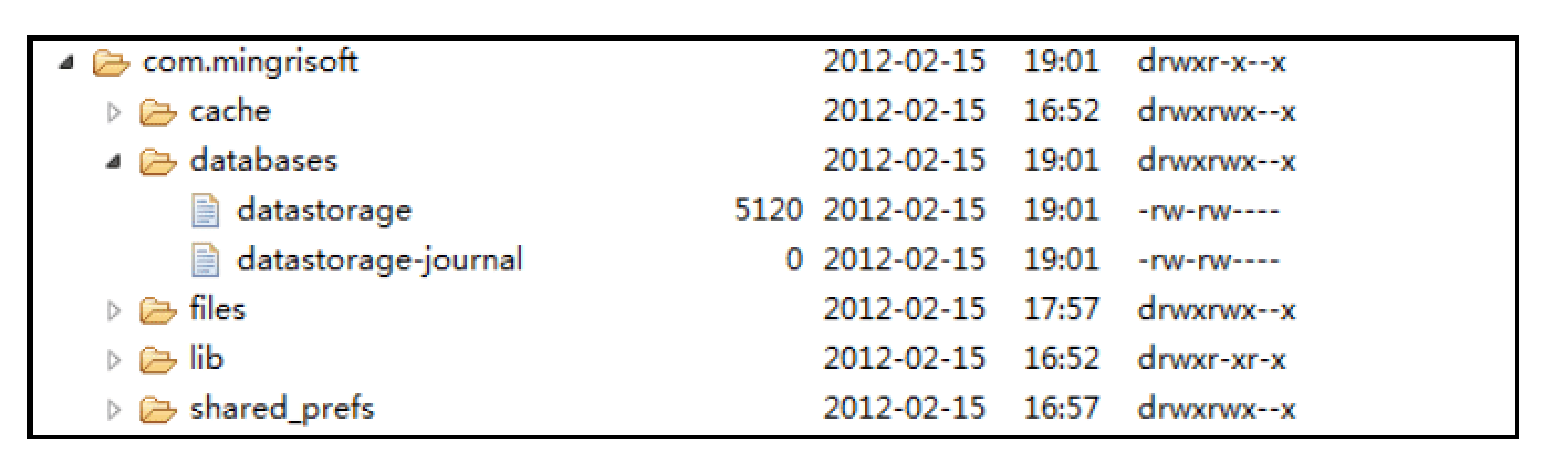


图 12.12　数据库文件保存位置

技巧：

在 Android 安装路径 tools 包中，提供了一个 sqlite3 命令工具，它可以用来操作 SQLite 数据库。例如，将图 12.12 中 datastorage 文件导出到 D 盘，启动 DOS 窗口，运行“sqlite3 d:\datastorage”命令，会显示如图 12.13 所示的提示信息。

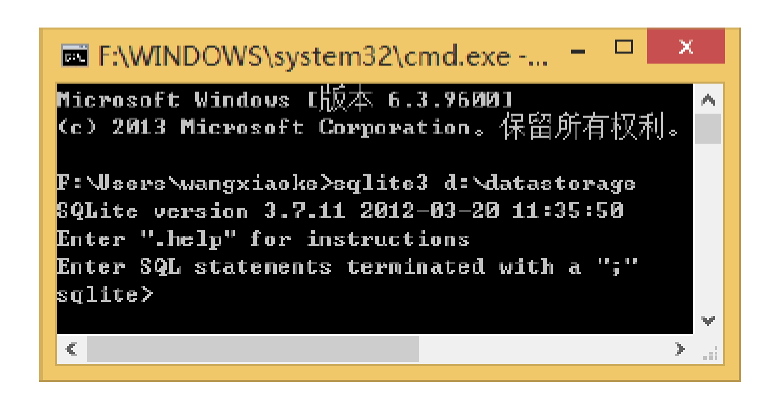


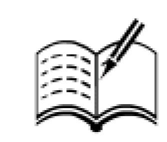
图 12.13　进入 sqlite 命令





注意：

如果不能显示如图 12.13 所示的信息，请将 sqlite3 命令所在的位置添加到系统环境变量中。



Note

12.4　综 合 应 用

12.4.1　遍历 Android 模拟器的 SD 卡

【例 12.6】 本实例主要实现使用列表显示 Android 模拟器 SD 卡上文件和文件夹名称的功能，运行程序，显示如图 12.14 所示文件和文件夹名称。



图 12.14　文件和文件夹名称列表

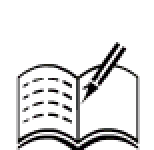
实例位置：光盘\MR\Instance\12\12.6

本实例实现时，首先需要使用 Environment.getExternalStorageDirectory()方法遍历 SD 卡中的所有目录，并通过 for 循环将遍历到的文件及文件夹名称存储到 List 泛型集合中，然后借助 ArrayAdapter 对象显示在 ListView 列表中。程序的开发步骤如下：

（1）修改 res\layout 文件夹中的 main.xml 文件，在该文件中定义一个 ListView 控件并修改它的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/list"
```





Note

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:dividerHeight="3dp"
            android:footerDividersEnabled="false"
            android:headerDividersEnabled="false" >
        </ListView>
</LinearLayout>
```

（2）在 res\layout 文件夹中创建 list_item.xml 文件，用来定义列表项的显示方式。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/row"
    android:layout_width="wrap_content"
    android:layout_height="25dp"
    android:textSize="20dp" />
```

（3）创建 FileListActivity 类，重写 onCreate()方法，使用 getExternalStorageDirectory()方法获得 SD 卡根路径，使用列表显示 SD 卡上文件和文件夹名称。其代码如下：

```
public class FileListActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                            //调用父类方法
        setContentView(R.layout.main);                                 //应用布局文件
        ListView lv = (ListView) findViewById(R.id.list);              //获得列表视图
        File rootPath = Environment.getExternalStorageDirectory(); //获得 SD 卡根路径
        List<String> items = new ArrayList<String>();                  //创建列表保存文件和文件夹名称
        for (File file : rootPath.listFiles()) {
            items.add(file.getName());                                 //遍历 SD 卡获得名称
        }
        ArrayAdapter<String> fileList = new ArrayAdapter<String>(this, R.layout.list_item, items);
        lv.setAdapter(fileList);                                       //设置列表适配器
    }
}
```

12.4.2 在 SQLite 数据库中批量添加数据

【例 12.7】 本实例主要实现向 SQLite 数据库中批量添加数据的功能，运行该程序之后，使用 DDMS 视图将 SQLite 数据库文件导出到 D 盘，使用 sqlite3 命令查看数据库文件的内容，如图 12.15 所示。

实例位置：光盘\MR\Instance\12\12.7

向 SQLite 数据库中批量添加数据时，主要借助在\res\raw 包中创建的 data 文件，该文件中保存了数字 1～9 的平方值和立方值。具体实现时，首先需要逐行遍历 data 文件的内容，然后使用 SQLiteDatabase 对象的 insert()方法向 SQLite 数据库中添加数据。程序的开发步骤如下：





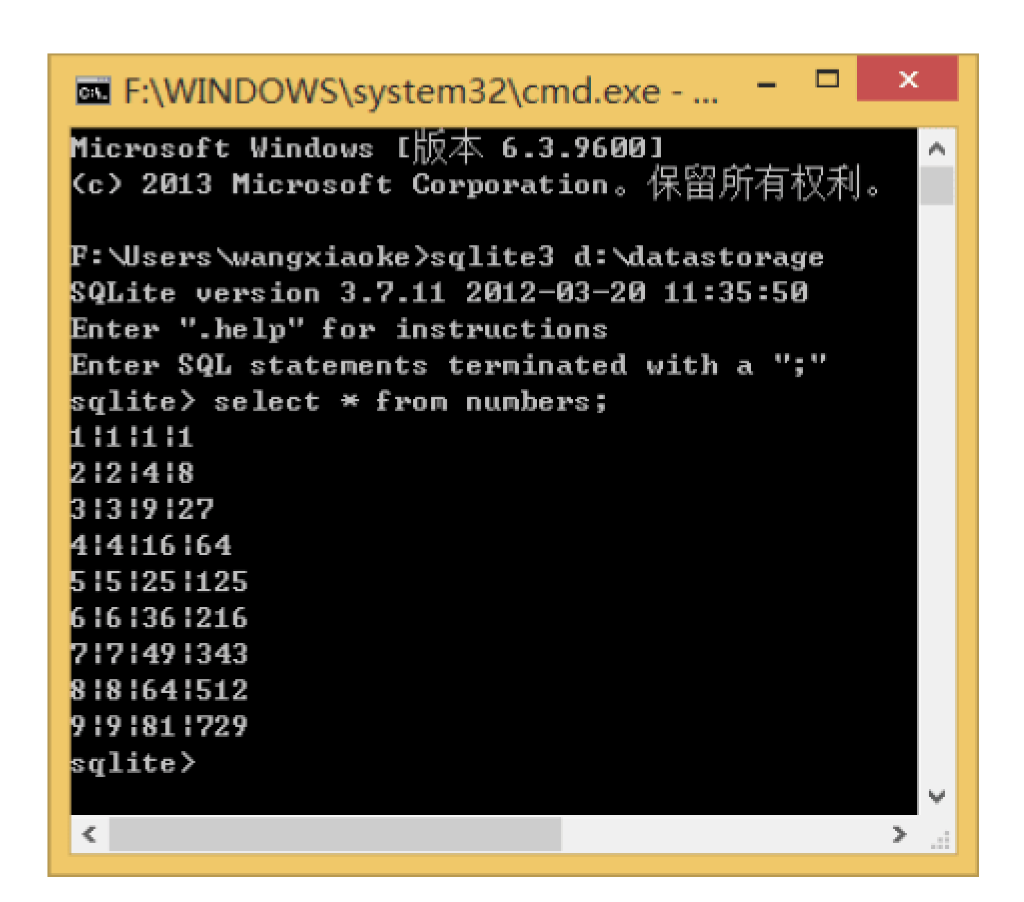


图 12.15　数据库文件中保存的数据

（1）在 com.mingrisoft.util 包中创建 DataBean 类，用来封装数字信息。其代码如下：

```
public class DataBean {
    private int id;
    private int number;
    private int square;
    private int cube;
    public DataBean() {
    }
    public int getId() {
        return id;
    }
    public int getNumber() {
        return number;
    }
    public void setNumber(int number) {
        this.number = number;
    }
    public int getSquare() {
        return square;
    }
    public void setSquare(int square) {
        this.square = square;
    }
    public int getCube() {
        return cube;
    }
    public void setCube(int cube) {
        this.cube = cube;
    }
}
```

（2）在 com.mingrisoft.util 包中创建 DBHelper 类，其中定义了若干字段来保存与数据库相关的信息。DBOpenHelper 类继承了 SQLiteOpenHelper 类，它提供了创建表格的功能。insert()方





法用于向数据库表格中保存数据。其代码如下：

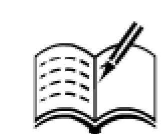
```
public class DBHelper {
    private static final String DATABASE_NAME = "datastorage";          //保存数据库名称
    private static final int DATABASE_VERSION = 1;                      //保存数据库版本号
    private static final String TABLE_NAME = "numbers";                 //保存表名称
    private static final String[] COLUMNS = { "_id", "number", "square", "cube" };
    private DBOpenHelper helper;
    private SQLiteDatabase db;
    private static class DBOpenHelper extends SQLiteOpenHelper {
        private static final String CREATE_TABLE = "create table " + TABLE_NAME + " ( " +
COLUMNS[0] + " integer primary key autoincrement, " + COLUMNS[1] + " integer, " + COLUMNS[2] + "
integer, " + COLUMNS[3] + " integer);";                                  //定义创建表格的 SQL 语句
        public DBOpenHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            db.execSQL(CREATE_TABLE);                                   //创建表格
        }
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            db.execSQL("drop table if exists " + TABLE_NAME);           //删除旧版表格
            onCreate(db);                                               //创建表格
        }
    }
    public DBHelper(Context context) {
        helper = new DBOpenHelper(context);                             //创建 SQLiteOpenHelper 对象
        db = helper.getWritableDatabase();                              //获得可写的数据库
    }
    public void insert(DataBean data) {                                 //向表格中插入数据
        ContentValues values = new ContentValues();
        values.put(COLUMNS[1], data.getNumber());
        values.put(COLUMNS[2], data.getSquare());
        values.put(COLUMNS[3], data.getCube());
        db.insert(TABLE_NAME, null, values);
    }
}
```

（3）创建 SQLiteWriteActivity 类，重写 onCreate()方法，获得 data 文件中的数据，然后将其保存到 SQLite 数据库中。其代码如下：

```
public class SQLiteWriteActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                             //调用父类方法
        setContentView(R.layout.main);                                  //应用布局文件
        DBHelper helper = new DBHelper(SQLiteWriteActivity.this);
        InputStream is = getResources().openRawResource(R.raw.data);  //获得输入流
        Scanner scanner = new Scanner(is);
```



```
            while (scanner.hasNextLine()) {
                String line = scanner.nextLine();                          //获得一行数据
                String[] data = line.split(" ");                           //使用空格将数据分行
                DataBean db = new DataBean();
                db.setNumber(Integer.parseInt(data[0]));                   //设置 number 值
                db.setSquare(Integer.parseInt(data[1]));                   //设置 square 值
                db.setCube(Integer.parseInt(data[2]));                     //设置 cube 值
                helper.insert(db);                                         //向数据库中插入一条数据
            }
        }
    }
```



Note

12.4.3　使用列表显示数据表中全部数据

【例 12.8】 本实例主要实现使用列表显示数据库中所有数据的功能，运行程序，效果如图 12.16 所示。

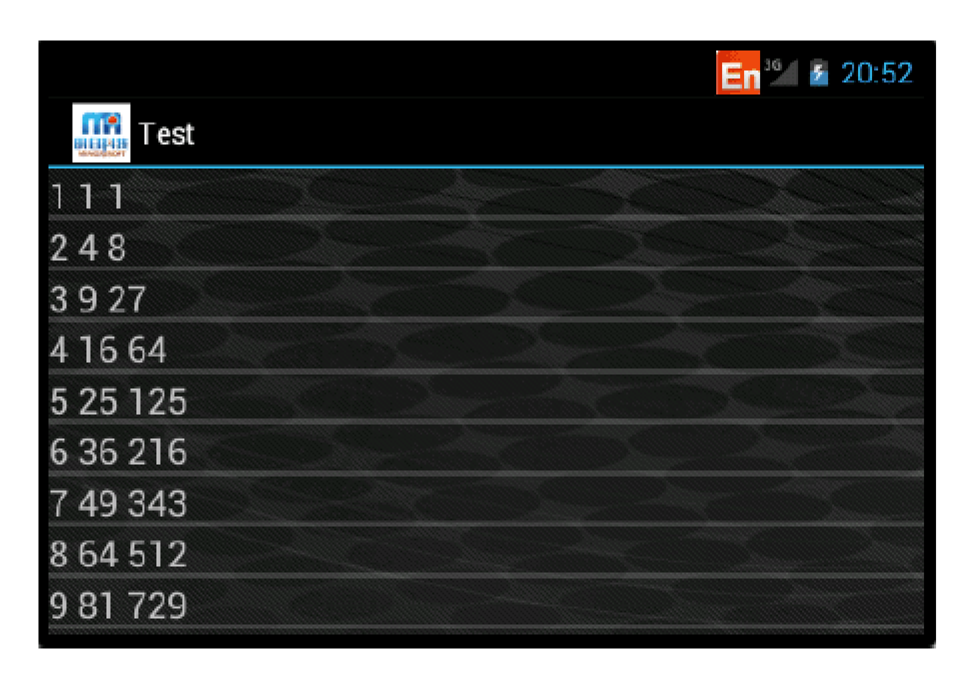


图 12.16　使用列表显示数据表中数据

实例位置：光盘\MR\Instance\12\12.8

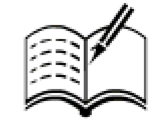
程序的开发步骤如下：

（1）修改 res\layout 文件夹中的 main.xml 文件，在该文件中定义一个 ListView 控件并修改它的默认属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <ListView
        android:id="@+id/list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:dividerHeight="3dp"
        android:footerDividersEnabled="false"
```



```
        android:headerDividersEnabled="false" >
    </ListView>
</LinearLayout>
```



Note

（2）在 res\layout 文件夹中创建 list_item.xml 文件，用来定义列表项的显示方式。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/row"
    android:layout_width="wrap_content"
    android:layout_height="25dp"
    android:textSize="20dp" />
```

（3）创建 DBHelper 类，该类中定义一个 queryAll()方法，用来获取数据表中的所有数据，并存储在 List 列表中。其代码如下：

```
public List<String> queryAll() {
    List<String> result = new ArrayList<String>();
    Cursor cursor = db.query(TABLE_NAME, COLUMNS, null, null, null, null, null);
    while (cursor.moveToNext()) {
        result.add(cursor.getInt(1) + " " + cursor.getInt(2) + " " + cursor.getInt(3));
    }
    return result;
}
```

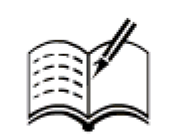
（4）创建 QueryActivity 类，重写 onCreate()方法，该方法中，调用 DBHelper 类中的 queryAll()方法获取数据表中的所有数据，并以列表的形式进行显示。其代码如下：

```
public class QueryActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        DBHelper helper = new DBHelper(this);
        ListView lv = (ListView) findViewById(R.id.list);          //获得列表视图
        ArrayAdapter<String> fileList = new ArrayAdapter<String>(this, R.layout.list_item, helper.queryAll());
        lv.setAdapter(fileList);                                    //设置列表适配器
    }
}
```

12.5　本章常见错误

运行使用 SQLite 数据库的 Android 程序时，出现下面的错误提示。






```
AndroidRuntime(5193):android.database.sqlite.SQLiteException:near "autoincreament":syntax error:create
table student(_id integer primary key autoincreament,name text,sex text,description text)
```

根据上面的异常信息可以发现，该错误是 SQLite 的语法错误，首先可以看到是 autoincreament 关键字附近出现了错误，然后可以将 create table student(_id integer primary key autoincreament, name text,sex text,description text)这条 SQL 语句复制到 SQLite 编译器中执行，具体查看一下这条 SQL 语句的出错位置，最后再根据这个错误修改 SQL 语句即可。

12.6　本章小结

本章主要向读者介绍了 Android 中的数据存储技术，常有的存储技术包括 Shared Preferences、Files 和 SQLite Databases，其中，Shared Preferences 适合存储简单的数据，如整数、布尔值等；Files 适合存储私有的数据及 SD 卡数据；SQLite Databases 适合存储复制的数据，它是一种轻便的数据库。数据存储技术在开发 Android 应用中经常用到，读者一定要熟练掌握。

12.7　跟我上机

参考答案：光盘\MR\跟我上机

在 Eclipse 中创建 Android 项目，主要实现复制图片到 SD 卡的功能。具体实现时，需要使用 getExternalStorageDirectory()方法获得 SD 卡根文件夹，然后需要使用文件流技术将制定的背景图片（Background.png）复制到 SD 卡上。其参考代码如下：

```
public class FileCopyActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);                         //调用父类方法
        setContentView(R.layout.main);                              //应用默认布局文件
        //创建文件对象
        File file = new File(Environment.getExternalStorageDirectory(), "Background.png");
        //打开输入流
        InputStream is = getResources().openRawResource(R.drawable.background);
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(file);                       //打开文件输出流
            byte[] buffer = new byte[is.available()];               //定义保存数据的数组
            is.read(buffer);                                        //从源文件中读取数据
            fos.write(buffer);                                      //将数据写入到新文件
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
```



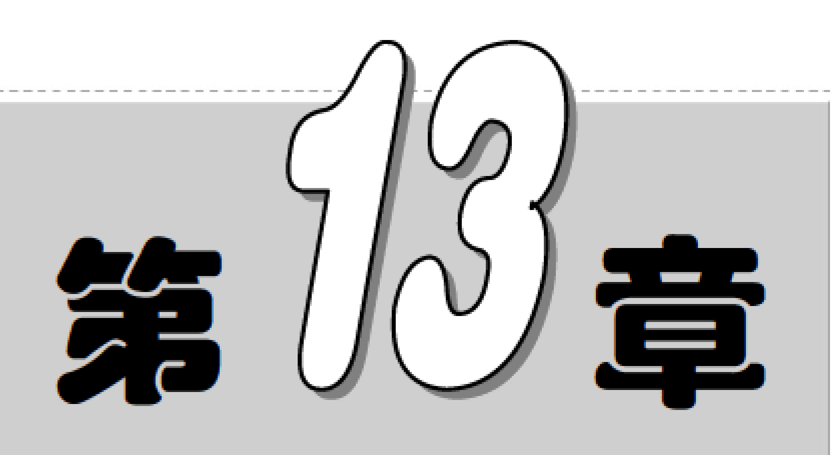
Note

```
            } finally {
                if (fos != null) {
                    try {
                        fos.close();                                    //关闭文件输出流
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
                if (is != null) {
                    try {
                        is.close();                                     //关闭输入流
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
        }
    }
```

说明：

向 SD 卡上复制图片时，需要增加外部存储写入权限，因此需要在 AndroidManifest.xml 配置文件中添加如下代码：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```



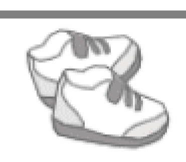
Content Provider 实现数据共享

（视频讲解：44 分钟）

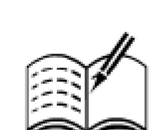
Content Provider 保存和获取数据并使其对所有应用程序可见。这是不同应用程序间共享数据的唯一方式。在 Android 中，没有提供所有应用共同访问的公共存储区域。本章将介绍如何使用预定义和自定义 Content Provider。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 使用 Content Provider 查询数据
- ☐ 使用 Content Provider 添加记录
- ☐ 使用 Content Provider 批量更新记录
- ☐ 使用 Content Provider 删除记录
- ☐ 自定义一个 Content Provider
- ☐ 查询联系人的 ID 和姓名
- ☐ 自动补全联系人姓名
- ☐ 显示联系人姓名和电话



13.1 Content Provider 概述



Content Provider 内部如何保存数据由其设计者决定，但是所有的 Content Provider 都实现一组通用的方法用来提供数据的增、删、改、查功能。

客户端通常不会直接使用这些方法，大多数是通过 ContentResolver 对象实现对 Content Provider 的操作。开发人员可以通过调用 Activity 或者其他应用程序组件的实现类中的 getContentResolver()方法来获得 ContentProvider 对象。例如：

```
ContentResolver cr = getContentResolver();
```

使用 ContentResolver 提供的方法可以获得 Content Provider 中任何感兴趣的数据。

当开始查询时，Android 系统确认查询的目标 Content Provider 并确保它正在运行。系统会初始化所有 ContentProvider 类的对象，开发人员不必完成此类操作。实际上，开发人员根本不会直接使用 ContentProvider 类的对象。通常，每个类型的 ContentProvider 仅有一个单独的实例。但是该实例能与位于不同应用程序和进程的多个 ContentResolver 类对象通信。不同进程之间的通信由 ContentProvider 类和 ContentResolver 类处理。

13.1.1 数据模型

Content Provider 使用基于数据库模型的简单表格来提供其中的数据，这里每行代表一条记录，每列代表特定类型和含义的数据。例如，联系人的信息可能以表 13.1 方式提供。

表 13.1 联系方式

_ID	NAME	NUMBER	EMAIL
001	张 XX	123*****	123**@163.com
002	王 XX	132*****	132**@google.com
003	李 XX	312*****	312**@qq.com
004	赵 XX	321*****	321**@126.com

每条记录包含一个数值型的_ID 字段，它用于在表格中唯一标识该记录。ID 能用于匹配相关表格中的记录，例如，在一个表格中查询联系人电话话，在另一表格中查询其照片。

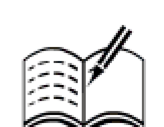


注意：

ID 字段前还包含了一个下划线，请读者在编写代码时不要忘记。

查询返回一个 Cursor 对象，它能遍历各行各列来读取各个字段的值。对于各个类型的数据，它都提供了专用的方法。因此，为了读取字段的数据，开发人员必须知道当前字段包含的数据类型。





13.1.2　URI 的用法

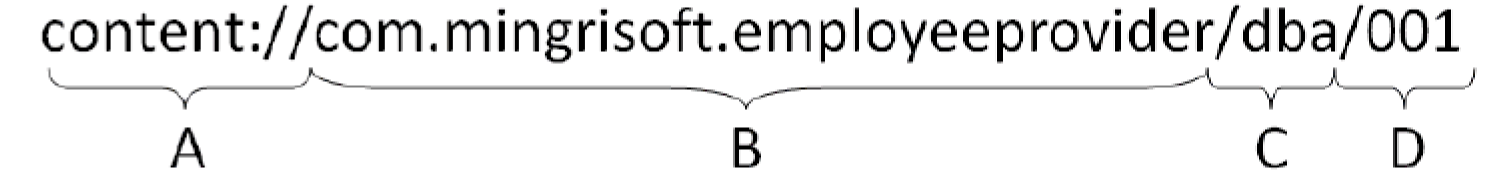
每个 Content Provider 提供公共的 URI（使用 Uri 类包装）来唯一标识其数据集。管理多个数据集（多个表格）的 Content Provider 为每个都提供了单独的 URI。所有为 provider 提供的 URI 都以“content://”作为前缀，“content://”模式表示数据由 Content Provider 来管理。

如果自定义 Content Provider，则应该为其 URI 也定义一个常量来简化客户端代码并让日后更新更加简洁。Android 为当前平台提供的 Content Provider 定义了 CONTENT_URI 常量。匹配电话号码到联系人表格的 URI 和匹配保存联系人照片表格的 URI 分别如下：

```
android.provider.Contacts.Phones.CONTENT_URI
android.provider.Contacts.Photos.CONTENT_URI
```

URI 常量用于所有与 Content Provider 的交互中。每个 ContentResolver 方法使用 URI 作为其第一个参数。它标识 ContentResolver 应该使用哪个 provider 及其中的哪个表格。

下面是 Content URI 重要部分的总结。



- ☑ A：标准的前缀，用于标识该数据由 Content Provider 管理。它永远不用修改。
- ☑ B：URI 的 authority 部分，它标识该 Content Provider。对于第三方应用，该部分应该是完整的类名（使用小写形式）来保证唯一性。在<provider>元素的 authorities 属性中声明 authority。
- ☑ C：Content Provider 的路径部分，用于决定哪类数据被请求。如果 Content Provider 仅提供一种数据类型，这部分可以没有。如果 provider 提供几种类型，包括子类型，这部分可以由几部分组成。
- ☑ D：被请求的特定记录的 ID 值。这是被请求记录的_ID 值。如果请求不仅限于单条记录，该部分及其前面的斜线应该删除。

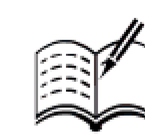
```
content://com.mingrisoft.employeeprovider/dba
```

13.2　Content Provider 的常用操作

Android 系统为常用数据类型提供了很多预定义的 Content Provider（声音、视频、图片和联系人等），它们大都位于 android.provider 包中。开发人员可以查询这些 provider 以获得其中包含的信息（尽管有些需要适当的权限来读取数据）。Android 系统提供的常见 Content Provider 说明如下。

- ☑ Browser：读取或修改书签、浏览历史或网络搜索。





- ☑ CallLog：查看或更新通话历史。
- ☑ Contacts：获取、修改或保存联系人信息。
- ☑ LiveFolders：由 ContentProvider 提供内容的特定文件夹。
- ☑ MediaStore：访问声音、视频和图片。
- ☑ Setting：查看和获取蓝牙设置、铃声和其他设备偏好。
- ☑ SearchRecentSuggestions：能被配置以使用查找意见 provider 操作。
- ☑ SyncStateContract：用于使用数据数组账号关联数据的 ContentProvider 约束。希望使用标准方式保存数据的 provider 可以使用它。
- ☑ UserDictionary：在可预测文本输入时，提供用户定义单词给输入法使用。应用程序和输入法能增加数据到该字典。单词能关联频率信息和本地化信息。

13.2.1　查询数据

开发人员需要下面 3 条信息才能查询 Content Provider 中的数据。

- ☑ 标识该 Content Provider 的 URI。
- ☑ 需要查询的数据字段名称。
- ☑ 字段中数据的类型。

如果查询特定的记录，则还需要提供该记录的 ID 值。

为了查询 Content Provider 中的数据，开发人员需要使用 ContentResolver.query()或 Activity.managedQuery()方法。这两个方法使用相同的参数，并且都返回 Cursor 对象。然而，managedQuery()方法导致 Activity 管理 Cursor 的生命周期。托管的 Cursor 处理所有的细节，如当 Activity 暂停时卸载自身和当 Activity 重启时加载自身。调用 Activity.startManagingCursor()方法可以让 Activity 管理未托管的 Cursor 对象。

query()和 managedQuery()方法的第一个参数是 provider 的 URI，即标识特定 ContentProvider 和数据集的 CONTENT_URI 常量。

为了限制仅返回一条记录，可以在 URI 结尾增加该记录的_ID 值，即将匹配 ID 值的字符串作为 URI 路径部分的结尾片段。例如，ID 值是 10，URI 将是：

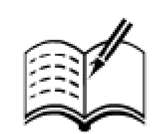
```
content://.../10
```

有些辅助方法，特别是 ContentUris.withAppendedId()和 Uri.withAppendedPath()，能轻松地将 ID 增加到 URI。这两个方法都是静态方法并返回一个增加了 ID 的 Uri 对象。

query()和 managedQuery()方法的其他参数用来更加细致地限制查询结果，它们是：

- ☑ 应该返回的数据列名称。null 值表示返回全部列。否则，仅返回列出的列。全部预定义 Content Provider 为其列都定义了常量。例如 android.provider.Contacts.Phones 类定义了_ID、NUMBER、NUMBER_KEY、NAME 等常量。
- ☑ 决定哪些行被返回的过滤器，格式类似 SQL 的 WHERE 语句（但是不包含 WHERE 自身）。null 值表示返回全部行（除非 URI 限制查询结果为单行记录）。
- ☑ 选择参数。





☑ 返回记录的排序器，格式类似 SQL 的 ORDER BY 语句（但是不包含 ORDER BY 自身）。null 值表示以默认顺序返回记录，这可能是无序的。

☑ 查询返回一组零条或多条数据库记录。列名、默认顺序和数据类型对每个 Content Provider 都是特别的。但是每个 provider 都有一个_ID 列，它为每条记录保存唯一的数值 ID。每个 provider 也能使用_COUNT 报告返回结果中记录的行数，该值在各行都是相同的。

☑ 获得数据使用 Cursor 对象处理，它能向前或者向后遍历整个结果集。开发人员可以使用它来读取数据。增加、修改和删除数据则必须使用 ContentResolver 对象。

13.2.2 增加记录

为了向 Content Provider 中增加新数据，首先需要在 ContentValues 对象中建立键值对映射，这里每个键匹配 content provider 中列名，每个值是该列中希望增加的值，然后调用 ContentResolver.insert()方法并传递给它 provider 的 URI 参数和 ContentValues 映射。该方法返回新记录的完整 URI，即增加了新记录 ID 的 URI。开发人员可以使用该 URI 来查询并获取该记录的 Cursor，以便修改该记录。

13.2.3 增加新值

一旦记录存在，开发人员可以向其增加新信息或者修改已经存在的信息。增加记录到 Contacts 数据库的最佳方式是增加保存新数据的表名到代表记录的 URI，然后使用组装好的 URI 来增加新数据。每个 Contacts 表格以 CONTENT_DIRECTORY 常量的方式提供名称作为该用途。

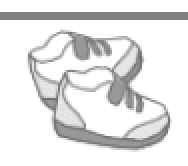
开发人员可以调用使用 byte 数组作为参数的 ContentValues.put()方法向表格中增加少量二进制数据。这适用于类似小图标的图片、短音频片段等。然而，如果需要增加大量二进制数据，如图片或者完整的歌曲，保存代表数据的 content:URI 到表格，然后使用文件 URI 调用 ContentResolver.openOutputStream()方法。这导致 Content Provider 保存数据到文件并在记录的隐藏字段保存文件路径。

13.2.4 批量更新记录

为了批量更新数据（如将全部字段中“NY”替换成“New York”），使用 ContentResolver.update()方法并提供需要修改的列名和值。

13.2.5 删除记录

如果需要删除单条记录，调用 ContentResolver.delete()方法并提供特定行的 URI。



如果需要删除多条记录，调用 ContentResolver.delete()方法并提供删除记录类型的 URI（如 android.provider.Contacts.People.CONTENT_URI）和一个 SQL WHERE 语句，它定义哪些行需要删除。



Note

注意：
请确保提供了一个合适的 WHERE 语句，否则可能删除全部数据。

13.3 自定义 Content Provider

如果开发人员希望共享自己的数据，则有如下两个选择。

☑ 创建自定义的 Content Provider（一个 ContentProvider 类的子类）。
☑ 如果有预定义的 provider，管理相同的数据类型并且有写入权限，则可以向其中增加数据。

前面已经详细介绍了如何使用系统预定义的 Content Provider，下面将介绍如何自定义 Content Provider。

如果自定义 Content Provider，则开发人员需要完成以下操作。

☑ 建立数据存储系统。大多数 Content Provider 使用 Android 文件存储方法或者 SQLite 数据库保存数据，但是开发人员可以使用任何方式存储。Android 提供了 SQLiteOpenHelper 类帮助创建数据库，SQLiteDatabase 类管理数据库。
☑ 继承 ContentProvider 类来提供数据访问方式。
☑ 在应用程序的 AndroidManifest 文件中声明 Content Provider。

下面介绍后两个任务。

13.3.1 继承 ContentProvider 类

开发人员定义 ContentProvider 类的子类，以便使用 ContentResolver 和 Cursor 类带来的便捷来共享数据。原则上，这意味着需要实现 ContentProvider 类定义的以下 6 个抽象方法。

```
public boolean onCreate()
public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)
public Uri insert(Uri uri, ContentValues values)
public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
public int delete(Uri uri, String selection, String[] selectionArgs)
public String getType(Uri uri)
```

各个方法的说明如表 13.2 所示。



Note

表 13.2　ContentProvider 中方法的抽象说明

方　法	说　明
onCreate()	用于初始化 provider
query()	返回数据给调用者
insert()	插入新数据到 Content Provider
update()	更新 Content Provider 中已经存在的数据
delete()	从 Content Provider 中删除数据
getType()	返回 Content Provider 数据的 MIME 类型

query()方法必须返回 Cursor 对象，用于遍历查询结果。Cursor 自身是一个接口，但是 Android 提供了一些该接口的实现类，例如，SQLiteCursor 能遍历存储在 SQLite 数据库中的数据。通过调用 SQLiteDatabase 类的 query()方法可以获得 Cursor 对象。它们都位于 android.database 包中，其继承关系如图 13.2 所示。

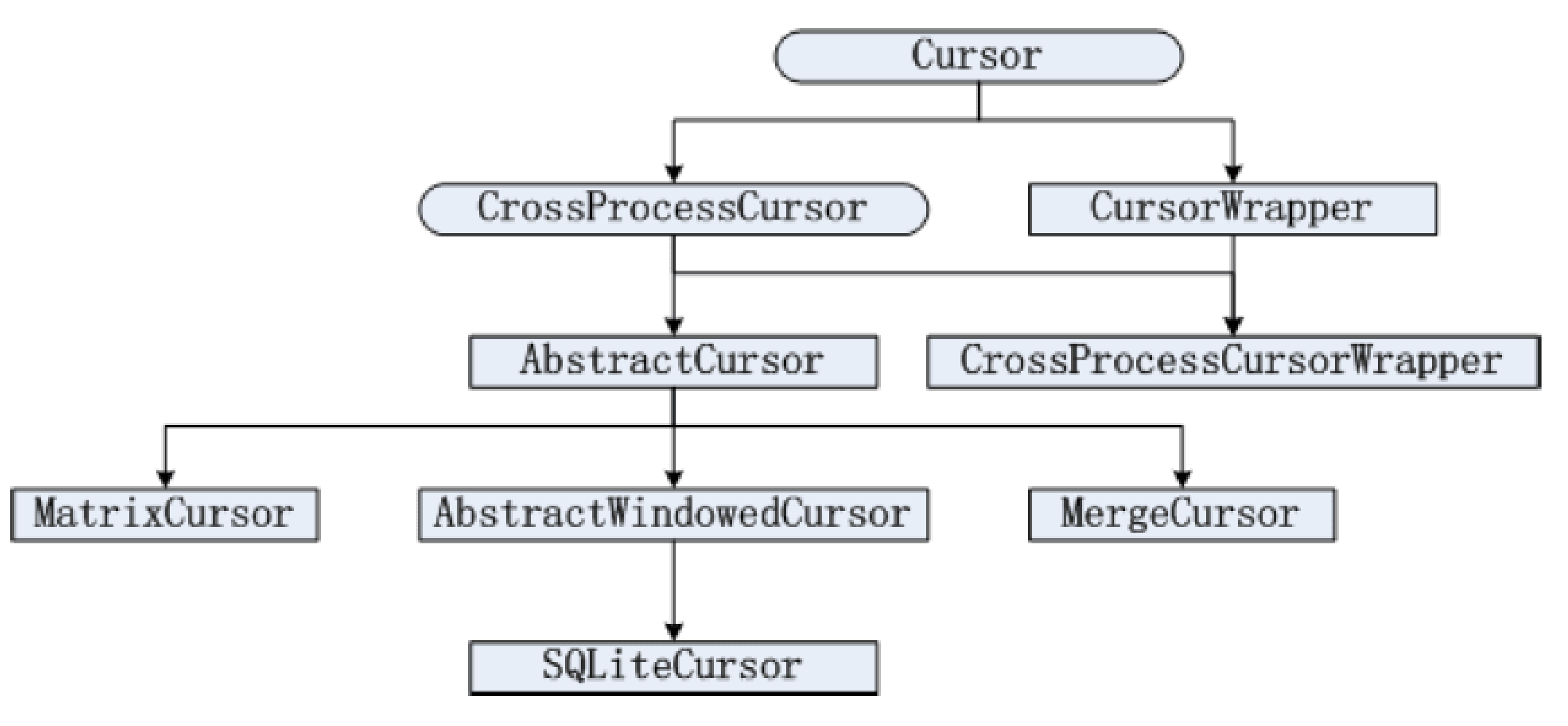


图 13.1　Cursor 接口继承关系



说明：

圆角矩形表示接口，非圆角矩形表示类。

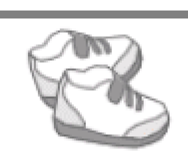
由于这些 ContentProvider 方法能被位于不同进程和线程的不同 ContentResolver 对象调用，它们必须以线程安全的方式实现。

此外，开发人员可能也想调用 ContentResolver.notifyChange()方法以便在数据修改时通知监听器。

除了定义子类自身，还应采取一些其他措施以便简化客户端工作并让类更加易用。

（1）定义 public static final Uri CONTENT_URI 变量（CONTENT_URI 是变量名称）。该字符串表示自定义的 Content Provider 处理的完整 content:URI。开发人员必须为该值定义唯一的字符串。最佳的解决方式是使用 Content Provider 的完整类名（小写）。例如，EmployeeProvider 的 URI 可能按如下定义：

```
public static final Uri CONTENT_URI = Uri.parse("content://com.mingrisoft.employeeprovider");
```



Note

如果 provider 包含子表，也应该为各个子表定义 URI。这些 URI 应该有相同的 authority（因为它标识 Content Provider），然后使用路径进行区分。例如：

```
content://com.mingrisoft.employeeprovider/dba
content://com.mingrisoft.employeeprovider/programmer
content://com.mingrisoft.employeeprovider/ceo
```

（2）定义 Content Provider 将返回给客户端的列名。如果开发人员使用底层数据库，这些列名通常与 SQL 数据库列名相同。同样定义 public static String 常量，客户端用它们来指定查询中的列和其他指令。确保包含名为“_ID”的整数列用来作为记录的 ID 值。无论记录中其他字段是否唯一，如 URL，开发人员都应该包含该字段。如果打算使用 SQLite 数据库，_ID 字段应该是如下类型：

```
INTEGER PRIMARY KEY AUTOINCREMENT
```

（3）仔细注释每列的数据类型，客户端需要使用这些信息来读取数据。

（4）如果开发人员正在处理新数据类型，则必须定义新的 MIME 类型以便在 ContentProvider.getType()方法实现中返回。

（5）如果开发人员提供的 byte 数据太大而不能放到表格中，如 bitmap 文件，提供给客户端的字段应该包含 content:URI 字符串。

13.3.2　声明 Content Provider

为了让 Android 系统知道开发人员编写的 Content Provider，应该在应用程序的 Android Manifest.xml 文件中定义<provider>元素。没有在配置文件中声明的自定义 Content Provider 对于 Android 系统不可见。

name 属性的值是 ContentProvider 类的子类的完整名称。authorities 属性是 provider 定义的 content:URI 中 authority 部分。ContentProvider 的子类是 EmployeeProvider，<provider>元素应该如下：

```
<provider android:name="com.mingrisoft.EmployeeProvider"
            android:authorities="com.mingrisoft.employeeprovider"
            . . . />
</provider>
```



注意：

authorities 属性删除了 content:URI 中的路径部分。

其他 provider 属性能设置读写数据的权限，提供显示给用户的图标或文本，启用或禁用 provider 等。如果数据不需要在多个运行着的 Content Provider 间同步，则设置 multiprocess 为 true。这允许在各个客户端进程创建一个 provider 实例，从而避免执行 IPC。





13.4　综 合 应 用

13.4.1　查询联系人 ID 和姓名

【例 13.1】 在 Eclipse 中创建 Android 项目，实现查询当前联系人应用中联系人的 ID 和姓名，运行程序，效果如图 13.2 所示。



图 13.2　显示联系人的 ID 和姓名

实例位置：光盘\MR\Instance\13\13.1

程序的开发步骤如下：

（1）修改 res\layout\main.xml 文件，设置背景图片和标签属性。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/black"
        android:textSize="25dp" />
</LinearLayout>
```

（2）创建 RetrieveDataActivity 类，它继承了 Activity 类。在 onCreate()方法中获得布局文件中定义的标签，在自定义的 getQueryData()方法中获得查询数据。其代码如下：

```
public class RetrieveDataActivity extends Activity {
    private String[] columns = { Contacts._ID,                    //希望获得 ID 值
```




Note

```
            Contacts.DISPLAY_NAME,                          //希望获得姓名
    };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = (TextView) findViewById(R.id.result);          //获得布局文件中的标签
        tv.setText(getQueryData());                                  //为标签设置数据
    }
    private String getQueryData() {
        StringBuilder sb = new StringBuilder();                      //用于保存字符串
        ContentResolver resolver = getContentResolver();             //获得 ContentResolver 对象
        //查询记录
        Cursor cursor = resolver.query(Contacts.CONTENT_URI, columns, null, null, null);
        int idIndex = cursor.getColumnIndex(columns[0]);             //获得 ID 记录的索引值
        int displayNameIndex = cursor.getColumnIndex(columns[1]);    //获得姓名记录的索引值
        //迭代全部记录
        for (cursor.moveToFirst(); !cursor.isAfterLast(); cursor.moveToNext()) {
            int id = cursor.getInt(idIndex);
            String displayName = cursor.getString(displayNameIndex);
            sb.append(id + ": " + displayName + "\n");
        }
        cursor.close();                                              //关闭 Cursor
        return sb.toString();                                        //返回查询结果
    }
}
```

（3）在 AndroidManifest 文件中增加读取联系人记录的权限。其代码如下：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

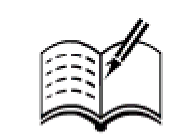
13.4.2　自动补全联系人姓名

【例 13.2】 本实例主要在 Android 程序中实现自动补全联系人姓名的功能，运行程序，效果如图 13.3 所示。



图 13.3　自动补全联系人姓名





实例位置：光盘\MR\Instance\13\13.2

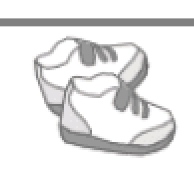
本实例实现自动补全联系人姓名时，主要是通过继承 CursorAdapter 类，并实现其 Filterable 接口实现的。创建 ContactListAdapter 类，它继承了 CursorAdapter 类并实现了 Filterable 接口，在重写方法时完成获取联系人姓名的功能。其代码如下：

```
public class ContactListAdapter extends CursorAdapter implements Filterable {
    private ContentResolver resolver;
    private   String[] columns = new String[] { Contacts._ID, Contacts.DISPLAY_NAME };
    public ContactListAdapter(Context context, Cursor c) {
        super(context, c);                          //调用父类构造方法
        resolver = context.getContentResolver();    //初始化 ContentResolver
    }
    @Override
    public void bindView(View arg0, Context arg1, Cursor arg2) {
        ((TextView) arg0).setText(arg2.getString(1));
    }
    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(context);
        TextView view = (TextView) inflater.inflate(android.R.layout.simple_dropdown_item_1line, parent,
false);
        view.setText(cursor.getString(1));
        return view;
    }
    @Override
    public CharSequence convertToString(Cursor cursor) {
        return cursor.getString(1);
    }
    @Override
    public Cursor runQueryOnBackgroundThread(CharSequence constraint) {
        FilterQueryProvider filter = getFilterQueryProvider();
        if (filter != null) {
            return filter.runQuery(constraint);
        }
        Uri uri = Uri.withAppendedPath(Contacts.CONTENT_FILTER_URI, Uri.encode(constraint.toString()));
        return resolver.query(uri, columns, null, null, null);
    }
}
```

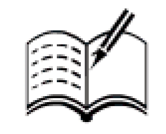
13.5　本章常见错误

在 Android 程序中使用 Content Provider 实现数据共享时，出现了下面的异常提示。

```
java.lang.SecurityException: Permission Denial: opening provider com.itcast.db.PersonProvider from ProcessRecord
{40f82218 23563:com.mingrisoft/u0a10079} (pid=23563,uid=10079) that is not exported from uid 10074
```



解决该异常，只需要在 AndroidManifest.xml 文件的 ContentProvider 定义中加上 android:exported="true"即可。



Note

13.6 本 章 小 结

本章重点介绍了 Android 中四大基本控件的 Content Provider，它是所有应用程序之间数据存储和检索的一个桥梁。在 Android 中，Content Provider 是一种特殊的数据存储类型，它提供了一套标准的方法来提供数据的增、删、改、查功能。本章详细介绍了实现各个功能需要使用的方法。此外，还介绍了如何自定义 Content Provider。

13.7 跟 我 上 机

参考答案：光盘\MR\跟我上机

创建一个 Android 程序，要求实现查询当前联系人应用中联系人姓名和电话的功能。具体实现时，首先需要获取读取联系人记录的权限，然后使用 ContentProvider 技术获取 Android 模拟器中存储的联系人姓名和电话，并显示在 TextView 控件中。

在 AndroidManifest 文件中获取读取联系人记录权限的代码如下：

```
<uses-permission android:name="android.permission.READ_CONTACTS"/>
```

创建 RetrieveDataActivity 类，继承自 Activity 类。在 onCreate()方法中获得布局文件中定义的标签，在自定义的 getQueryData()方法中查询数据，获取当前联系人应用中的联系人姓名和电话。其代码如下：

```
public class RetrieveDataActivity extends Activity {
    private String[] columns = { Contacts._ID,                          //获得 ID 值
            Contacts.DISPLAY_NAME,                                      //获得姓名
            Phone.NUMBER,                                               //获得电话
            Phone.CONTACT_ID, };
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView tv = (TextView) findViewById(R.id.result);             //获得布局文件中的标签
        tv.setText(getQueryData());                                     //为标签设置数据
    }
    private String getQueryData() {
        StringBuilder sb = new StringBuilder();                         //用于保存字符串
        ContentResolver resolver = getContentResolver();                //获得 ContentResolver 对象
        //查询记录
```






```
            Cursor cursor = resolver.query(Contacts.CONTENT_URI, null, null, null, null);
            while (cursor.moveToNext()) {
                int idIndex = cursor.getColumnIndex(columns[0]);                //获得 ID 值的索引
                int displayNameIndex = cursor.getColumnIndex(columns[1]);       //获得姓名索引
                int id = cursor.getInt(idIndex);                                //获得 id
                String displayName = cursor.getString(displayNameIndex);        //获得名称
                Cursor phone = resolver.query(Phone.CONTENT_URI, null, columns[3] + "=" + id, null, null);
                while (phone.moveToNext()) {
                    int phoneNumberIndex = phone.getColumnIndex(columns[2]);    //获得电话索引
                    String phoneNumber = phone.getString(phoneNumberIndex);     //获得电话
                    sb.append(displayName + ": " + phoneNumber + "\n");         //保存数据
                }
            }
            cursor.close();                                                     //关闭游标
            return sb.toString();
        }
}
```



图形图像处理技术

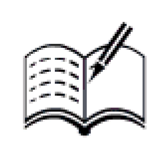
（视频讲解：2 小时 8 分钟）

图形图像处理技术在 Android 中非常重要，特别是在开发益智类游戏或者 2D 游戏时，都离不开图形图像处理技术的支持。本章将对 Android 中的图形图像处理技术进行详细介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- □ 绘制以渐变色填充的矩形
- □ 创建绘图画布并绘制带阴影的矩形
- □ 绘制 5 个不同颜色的圆形
- □ 在画布上绘制文字
- □ 绘制路径及绕路径文字
- □ 在屏幕上绘制 Android 的机器人
- □ 使用 Matrix 旋转、缩放、倾斜和平移图像
- □ 显示平铺背景和椭圆形的图片
- □ 旋转、平移、缩放和透明度渐变的补间动画
- □ 实现带描边的圆角图片
- □ 实现放大镜效果
- □ 忐忑的精灵





14.1　Android 中的常用绘图类

在 Android 中，绘制图像时，最常应用的就是 Paint、Canvas、Bitmap 和 BitmapFactory 类。其中 Paint 类代表画笔，Canvas 类代表画布。在实际生活中，有画笔和画布就可以正常作画了，在 Android 中，也是如此，通过 Paint 类和 Canvas 类就可绘制图像了。下面将对这 4 个类进行详细介绍。

14.1.1　Paint 类

Paint 类代表画笔，用来描述图形的颜色和风格，如线宽、颜色、透明度和填充效果等信息。使用 Paint 类时，需要先创建该类的对象，这可以通过该类提供的构造方法来实现。通常情况下，只需要使用 Paint()方法来创建一个使用默认设置的 Paint 对象。其具体代码如下：

```
Paint paint=new Paint();
```

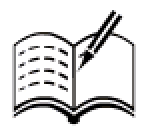
创建 Paint 类的对象后，还可以通过该对象提供的方法来对画笔的默认设置进行改变，例如，改变画笔的颜色、笔触宽度等。用于改变画笔设置的常用方法如表 14.1 所示。

表 14.1　Paint 类的常用方法

方　法	描　述
setARGB(int a, int r, int g, int b)	用于设置颜色，各参数值均为 0～255 之间的整数，分别用于表示透明度、红色、绿色和蓝色值
setColor(int color)	用于设置颜色，参数 color 可以通过 Color 类提供的颜色常量指定，也可以通过 Color.rgb(int red,int green,int blue)方法指定
setAlpha(int a)	用于设置透明度，值为 0～255 之间的整数
setAntiAlias(boolean aa)	用于指定是否使用抗锯齿功能，如果使用会使绘图速度变慢
setDither(boolean dither)	用于指定是否使用图像抖动处理，如果使用会使图像颜色更加平滑和饱满，更加清晰
setPathEffect(PathEffect effect)	用于设置绘制路径时的路径效果，如点画线
setShader(Shader shader)	用于设置渐变，可以使用 LinearGradient（线性渐变）、RadialGradient（径向渐变）或者 SweepGradient（角度渐变）
setShadowLayer(float radius, float dx, float dy, int color)	用于设置阴影，参数 radius 为阴影的角度，dx 和 dy 为阴影在 x 轴和 y 轴上的距离，color 为阴影的颜色。如果参数 radius 的值为 0，那么将没有阴影
setStrokeCap(Paint.Cap cap)	用于当画笔的填充样式为 STROKE 或 FILL_AND_STROKE 时，设置笔刷的图形样式，参数值可以是 Cap.BUTT、Cap.ROUND 或 Cap.SQUARE。主要体现在线的端点上
setStrokeJoin(Paint.Join join)	用于设置画笔转弯处的连接风格，参数值为 Join.BEVEL、Join.MITER 或 Join.ROUND
setStrokeWidth(float width)	用于设置笔触的宽度







Note

续表

方　法	描　述
setStyle(Paint.Style style)	用于设置填充风格，参数值为 Style.FILL、Style.FILL_AND_STROKE 或 Style.STROKE
setTextAlign(Paint.Align align)	用于设置绘制文本时的文字对齐方式，参数值为 Align.CENTER、Align.LEFT 或 Align.RIGHT
setTextSize(float textSize)	用于设置绘制文本时的文字的大小
setFakeBoldText(boolean fakeBoldText)	用于设置是否为粗体文字
setXfermode(Xfermode xfermode)	用于设置图形重叠时的处理方式，如合并、取交集或并集，经常用来制作橡皮的擦除效果

例如，要定义一个画笔，指定该画笔的颜色为绿色，带一个浅灰色的阴影，可以使用下面的代码。

```
Paint paint=new Paint();
paint.setColor(Color. RED);
paint.setShadowLayer(2, 3, 3, Color.rgb(180, 180, 180));
```

应用该画笔，在画布上绘制一个带阴影的矩形的效果如图 14.1 所示。

【例 14.1】 分别定义一个线性渐变、径向渐变和角度渐变的画笔，并应用这 3 支画笔绘制 3 个矩形。

实例位置：光盘\MR\Instance\14\14.1

其关键代码如下：

```
Paint paint=new Paint();                                    //定义一个默认的画笔
//线性渐变
Shader shader=new LinearGradient(0, 0, 50, 50, Color.RED, Color.GREEN, Shader.TileMode.MIRROR);
paint.setShader(shader);                                    //为画笔设置渐变器
canvas.drawRect(10, 70, 100, 150, paint);                   //绘制矩形
//径向渐变
shader=new RadialGradient(160, 110, 50, Color.RED, Color.GREEN, Shader.TileMode.MIRROR);
paint.setShader(shader);                                    //为画笔设置渐变器
canvas.drawRect(115,70,205,150, paint);                     //绘制矩形
//角度渐变
shader=new SweepGradient(265,110,new int[]{Color.RED,Color.GREEN,Color.BLUE},null);
paint.setShader(shader);                                    //为画笔设置渐变器
canvas.drawRect(220, 70, 310, 150, paint);                  //绘制矩形
```

运行本实例，将显示如图 14.2 所示的运行效果。



图 14.1　绘制带阴影的矩形

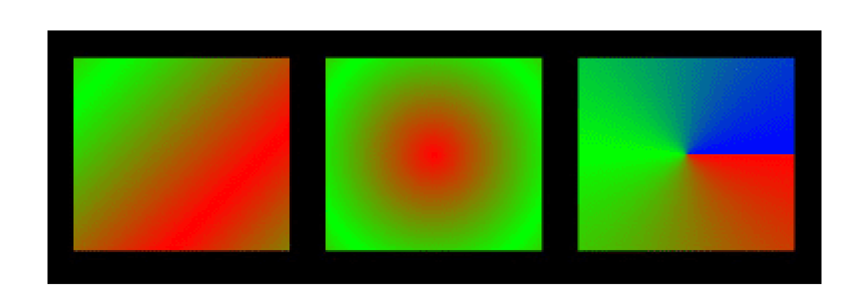
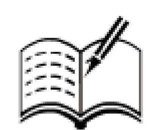


图 14.2　绘制以渐变色填充的矩形





14.1.2　Canvas 类

Canvas 类代表画布，通过该类提供的方法，可以绘制各种图形（如矩形、圆形和线条等）。通常情况下，要在 Android 中绘图，需要先创建一个继承自 View 类的视图，并且在该类中重写它的 onDraw(Canvas canvas)方法，然后在显示绘图的 Activity 中添加该视图。下面将通过一个具体的实例来说明如何创建用于绘图的画布。

【例 14.2】 在 Eclipse 中创建 Android 项目，实现创建绘图画布功能。

实例位置：光盘\MR\Instance\14\14.2

程序的开发步骤如下：

（1）创建一个名称为 DrawView 的类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法。其关键代码如下：

```
public class DrawView extends View {
	/**
	 * 功能：构造方法
	 */
	public DrawView(Context context, AttributeSet attrs) {
		super(context, attrs);
	}
	/*
	 * 功能：重写 onDraw()方法
	 */
	@Override
	protected void onDraw(Canvas canvas) {
		super.onDraw(canvas);
	}
}
```

说明：

上面加粗的代码为重写 onDraw()方法的代码。在重写的 onDraw()方法中，可以编写绘图代码，参数 canvas 就是要进行绘图的画布。

（2）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，并在帧布局管理器中添加步骤（1）创建的自定义视图。修改后的代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
    <com.mingrisoft.DrawView
```

```
        android:id="@+id/drawView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</FrameLayout>
```

（3）在 DrawView 的 onDraw()方法中，添加以下代码用于绘制一个带阴影的红色矩形。

```
Paint paint=new Paint();                                    //定义一个采用默认设置的画笔
paint.setColor(Color.RED);                                  //设置颜色为红色
paint.setShadowLayer(2, 3, 3, Color.rgb(180, 180, 180));    //设置阴影
canvas.drawRect(40, 40, 200, 100, paint);                   //绘制矩形
```

运行本实例，将显示如图 14.3 所示的运行效果。



图 14.3　创建绘图画布并绘制带阴影的矩形

14.1.3　Bitmap 类

Bitmap 类代表位图，它是 Android 系统中图像处理的最重要类之一，使用它不仅可以获取图像文件信息进行图像剪切、旋转、缩放等操作，而且还可以指定格式保存图像文件。Bitmap 类提供的常用方法如表 14.2 所示。

表 14.2　Bitmap 类的常用方法

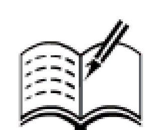
方　法	描　述
compress(Bitmap.CompressFormat format, int quality, OutputStream stream)	用于将 Bitmap 对象压缩为指定格式并保存到指定的文件输出流中，其中 format 参数值可以是 Bitmap.CompressFormat.PNG、Bitmap.CompressFormat. JPEG 和 Bitmap.CompressFormat.WEBP
createBitmap(Bitmap source, int x, int y, int width, int height, Matrix m, boolean filter)	用于从源位图的指定坐标点开始，“挖取”指定宽度和高度的一块图像来创建新的 Bitmap 对象，并按 Matrix 指定规则进行变换
createBitmap(int width, int height, Bitmap.Config config)	用于创建一个指定宽度和高度的新的 Bitmap 对象
createBitmap(Bitmap source, int x, int y, int width, int height)	用于从源位图的指定坐标点开始，“挖取”指定宽度、和高度的一块图像来创建新的 Bitmap 对象
createBitmap(int[] colors, int width, int height, Bitmap. Config config)	使用颜色数组创建一个指定宽度和高度的新 Bitmap 对象，其中，数组元素的个数为 width*height
createBitmap(Bitmap src)	用于使用源位图创建一个新的 Bitmap 对象
createScaledBitmap(Bitmap src, int dstWidth, int dstHeight, boolean filter)	用于将源位图缩放为指定宽度和高度的新的 Bitmap 对象
isRecycled()	用于判断 Bitmap 对象是否被回收
recycle()	强制回收 Bitmap 对象





说明：

表 14.2 中给出的方法不包括对图像进行缩放和旋转的方法，关于如何使用 Bitmap 类对图像进行缩放和旋转将在 14.3 节进行介绍。



Note

例如，创建一个包括 4 个像素（每个像素对应一种颜色）的 Bitmap 对象的代码如下：

```
Bitmap  bitmap=Bitmap.createBitmap(new  int[]{Color.RED,Color.GREEN,Color.BLUE,Color.MAGENTA},
4, 1, Config.RGB_565);
```

14.1.4　BitmapFactory 类

在 Android 中，还提供了一个 BitmapFactory 类，该类为一个工具类，用于从不同的数据源来解析、创建 Bitmap 对象。BitmapFactory 类提供的创建 Bitmap 对象的常用方法如表 14.3 所示。

表 14.3　BitmapFactory 类的常用方法

方　法	描　述
decodeFile(String pathName)	用于从给定的路径所指定的文件中解析、创建 Bitmap 对象
decodeFileDescriptor(FileDescriptor fd)	用于从 FileDescriptor 对应的文件中解析、创建 Bitmap 对象
decodeResource(Resources res, int id)	用于根据给定的资源 id 从指定的资源中解析、创建 Bitmap 对象
decodeStream(InputStream is)	用于从指定的输入流中解析、创建 Bitmap 对象

例如，要解析 SD 卡上的图片文件 img01.jpg，并创建对应的 Bitmap 对象可以使用下面的代码。

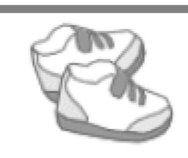
```
String path="/sdcard/pictures/bccd/img01.jpg";
Bitmap bm=BitmapFactory.decodeFile(path);
```

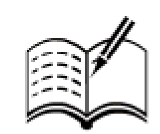
要解析 Drawable 资源中保存的图片文件 img02.jpg，并创建对应的 Bitmap 对象，可以使用下面的代码。

```
Bitmap bm=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.img02);
```

14.2　绘制 2D 图像

在 Android 中，提供了非常强大的本机二维图形库，用于绘制 2D 图像。在 Android 应用中，比较常用的是绘制几何图形、文本、路径和图片等。下面分别进行介绍。





Note

14.2.1 绘制几何图形

比较常见的几何图形包括点、线、弧、圆形、矩形等。在 Android 中，Canvas 类提供了丰富的绘制几何图形的方法，通过这些方法可以绘制出各种几何图形。常用的绘制几何图形的方法如表 14.4 所示。

表 14.4 Canvas 类提供的绘制几何图形的方法

方 法	描 述	举 例	绘图效果
drawArc(RectF oval, float startAngle, float sweepAngle, boolean useCenter, Paint paint)	绘制弧	RectF rectf=new RectF(10, 20, 100, 110); canvas.drawArc(rectf, 0, 60, true, paint);	
		RectF rectf1=new RectF(10, 20, 100, 110); canvas.drawArc(rectf1, 0, 60, false, paint);	
drawCircle(float cx, float cy, float radius, Paint paint)	绘制圆形	paint.setStyle(Style.STROKE); canvas.drawCircle(50, 50, 15, paint);	
drawLine(float startX, float startY, float stopX, float stopY, Paint paint)	绘制一条线	canvas.drawLine(100, 10, 150, 10, paint);	
drawLines(float[] pts, Paint paint)	绘制多条线	canvas.drawLines(new float[]{10,10, 30, 10, 30, 10, 15,30, 15,30, 10,10}, paint);	
drawOval(RectF oval, Paint paint)	绘制椭圆	RectF rectf=new RectF(40, 20, 80, 40); canvas.drawOval(rectf,paint);	
drawPoint(float x, float y, Paint paint)	绘制一个点	canvas.drawPoint(10, 10, paint);	
drawPoints(float[] pts, Paint paint)	绘制多个点	canvas.drawPoints(new float[]{10,10, 15, 10, 20,15, 25,10, 30,10}, paint);	
drawRect(float left, float top, float right, float bottom, Paint paint)	绘制矩形	canvas.drawRect(10, 10, 40, 30, paint);	
drawRoundRect(RectF rect, float rx, float ry, Paint paint)	绘制圆角矩形	RectF rectf=new RectF(40, 20, 80, 40); canvas.drawRoundRect(rectf, 6, 6, paint);	



说明：

表 14.4 中给出的绘图效果使用的画笔均为以下代码所定义的画笔。

```
Paint paint=new Paint();                //创建一个采用默认设置的画笔
paint.setAntiAlias(true);               //使用抗锯齿功能
paint.setColor(Color.RED);              //设置颜色为红色
paint.setStrokeWidth(2);                //笔触的宽度为 2 像素
paint.setStyle(Style.STROKE);           //填充样式为描边
```

【例 14.3】 在 Eclipse 中创建 Android 项目，实现绘制 5 个不同颜色的圆形组成的图案。

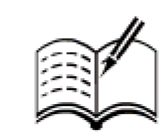
实例位置：光盘\MR\Instance\14\14.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器



和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。修改后的代码如下：




```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frameLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
</FrameLayout>
```

（2）打开默认创建的 MainActivity，在该文件中，创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法。关键代码如下：

```
public class MyView extends View{
    public MyView(Context context) {
        super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
    }
}
```

（3）在 MainActivity 的 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将步骤（2）中创建的 MyView 视图添加到该帧布局管理器中。其关键代码如下：

```
//获取布局文件中添加的帧布局管理器
FrameLayout ll=(FrameLayout)findViewById(R.id.frameLayout1);
ll.addView(new MyView(this));                    //将自定义的 MyView 视图添加到帧布局管理器中
```

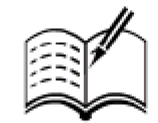
（4）在 DrawView 的 onDraw()方法中，首先指定画布的背景色，然后创建一个采用默认设置的画笔，并设置该画笔使用抗锯齿功能，以及笔触的宽度，再设置填充样式为描边，最后设置画笔颜色并绘制圆形。其具体代码如下：

```
canvas.drawColor(Color.WHITE);                  //指定画布的背景色为白色
Paint paint=new Paint();                        //创建采用默认设置的画笔
paint.setAntiAlias(true);                       //使用抗锯齿功能
paint.setStrokeWidth(3);                        //设置笔触的宽度
paint.setStyle(Style.STROKE);                   //设置填充样式为描边
paint.setColor(Color.BLUE);
canvas.drawCircle(50, 50, 30, paint);           //绘制蓝色的圆形
paint.setColor(Color.YELLOW);
canvas.drawCircle(100, 50, 30, paint);          //绘制黄色的圆形
paint.setColor(Color.BLACK);
canvas.drawCircle(150, 50, 30, paint);          //绘制黑色的圆形
paint.setColor(Color.GREEN);
canvas.drawCircle(75, 90, 30, paint);           //绘制绿色的圆形
```



```
paint.setColor(Color.RED);
canvas.drawCircle(125, 90, 30, paint);            //绘制红色的圆形
```

运行本实例，将显示如图 14.4 所示的运行效果。



Note

14.2.2　绘制文本

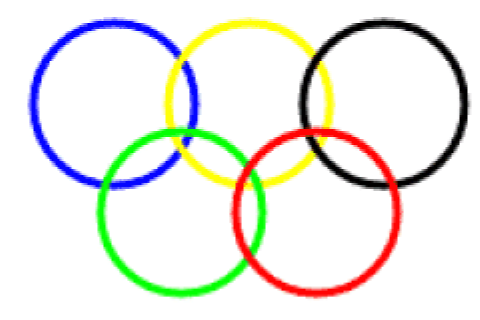


图 14.4　绘制 5 个不同颜色的圆形

在 Android 中，虽然可以通过 TextView 或是图片显示文本，但是在开发游戏时，特别是开发 RPG（角色）类游戏时，会包含很多文字，使用 TextView 和图片显示文本不太合适，这时就需要通过绘制文本的方式来实现。Canvas 类提供了一系列的绘制文本的方法，下面分别进行介绍。

1．drawText()方法

drawText()方法用于在画布的指定位置绘制文字。该方法比较常用的语法格式如下：

```
drawText(String text, float x, float y, Paint paint)
```

在该语法中，参数 text 用于指定要绘制的文字；参数 x 用于指定文字的起始位置的 x 轴坐标；参数 y 用于指定文字的起始位置的 y 轴坐标；参数 paint 用于指定使用的画笔。

例如，要在画布上输出文字“明日科技”可以使用下面的代码。

```
Paint paintText=new Paint();
paintText.setTextSize(20);
canvas.drawText("明日科技", 165,65, paintText);
```

2．drawPosText()方法

drawPosText()方法也是用于在画布上绘制文字，与 drawText()方法不同的是，使用该方法绘制字符串时，需要为每个字符指定一个位置。该方法比较常用的语法格式如下：

```
drawPosText(String text, float[] pos, Paint paint)
```

在该语法中，参数 text 用于指定要绘制的文字；参数 pos 用于指定每一个字符的位置；参数 paint 用于指定要使用的画笔。

例如，要在画布上分两行输出文字，可以使用下面的代码。

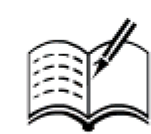
```
Paint paintText=new Paint();
paintText.setTextSize(24);
float[] pos= new float[]{80,215, 105,215, 130,215,80,240, 105,240, 130,240};
canvas.drawPosText("很高兴见到你", pos, paintText);
```

【例 14.4】 在 Eclipse 中创建 Android 项目，实现绘制一个游戏对白界面。

实例位置：光盘\MR\Instance\14\14.4

程序的开发步骤如下：





（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，并为其设置背景，用于显示自定义的绘图类。修改后的代码如下：

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/frameLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
</FrameLayout>
```

（2）打开默认创建的 MainActivity，在该文件中，创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法。其关键代码如下：

```
public class MyView extends View{
    public MyView(Context context) {
        super(context);
    }
    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
    }
}
```

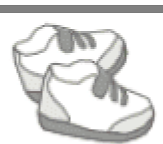
（3）在 MainActivity 的 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将步骤（2）中创建的 MyView 视图添加到该帧布局管理器中。其关键代码如下：

```
FrameLayout ll=(FrameLayout)findViewById(R.id.frameLayout1);
ll.addView(new MyView(this));                    //将自定义的 MyView 视图添加到帧布局管理器中
```

（4）在 MyView 的 onDraw()方法中，首先创建一个采用默认设置的画笔，然后设置画笔颜色，以及对齐方式、文字大小和使用抗锯齿功能，再通过 drawText()方法绘制一段文字，最后通过 drawPosText()方法绘制文字。其具体代码如下：

```
Paint paintText=new Paint();                     //创建一个采用默认设置的画笔
paintText.setColor(0xFFFF6600);                  //设置画笔颜色
paintText.setTextAlign(Align.LEFT);              //设置文字左对齐
paintText.setTextSize(24);                       //设置文字大小
paintText.setAntiAlias(true);                    //使用抗锯齿功能
canvas.drawText("不，我不想去！", 520,75, paintText);          //通过 drawText()方法绘制文字
//定义代表文字位置的数组
float[] pos= new float[]{400,260, 425,260, 450,260, 475,260,
    363,290, 388,290, 413,290, 438,290, 463,290, 488,290, 513,290};
canvas.drawPosText("你想和我一起去探险吗？", pos, paintText);//通过 drawPosText()方法绘制文字
```

运行本实例，将显示如图 14.5 所示的运行效果。



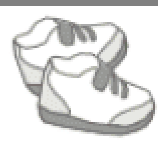
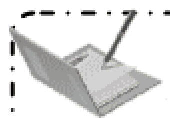




图 14.5　在画布上绘制文字



说明：

运行本实例时，需要将 Android 模拟器（AVD）修改为 WSVGA 模式。

14.2.3　绘制路径

在 Android 中提供了绘制路径的功能。绘制一条路径可以分为创建路径和绘制定义好的路径两部分，下面分别进行介绍。

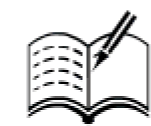
1．创建路径

要创建路径可以使用 android.graphics.Path 类来实现。Path 类包含一组矢量绘图方法，如画圆、矩形、弧、线条等。常用的绘图方法如表 14.5 所示。

表 14.5　Path 类的常用方法

方　　法	描　　述
addArc(RectF oval, float startAngle, float sweepAngle)	添加弧形路径
addCircle(float x, float y, float radius, Path.Direction dir)	添加圆形路径
addOval(RectF oval, Path.Direction dir)	添加椭圆形路径
addRect(RectF rect, Path.Direction dir)	添加矩形路径
addRoundRect(RectF rect, float rx, float ry, Path.Direction dir)	添加圆角矩形路径
moveTo(float x, float y)	设置开始绘制直线的起始点
lineTo(float x, float y)	在 moveTo()方法设置的起始点与该方法指定的结束点之间画一条直线，如果在调用该方法之前没使用 moveTo()方法设置起始点，那么将从（0,0）点开始绘制直线
quadTo(float x1, float y1, float x2, float y2)	用于根据指定的的参数绘制一条线段轨迹
close()	闭合路径







说明：

在使用 addCircle()、addOval()、addRect()和 addRoundRect()方法时，需要指定 Path.Direction 类型的常量，可选值为 Path.Direction.CW（顺时针）和 Path.Direction.CCW（逆时针）。

例如，要创建一个顺时针旋转的圆形路径可以使用下面的代码。

```
Path path=new Path();                          //创建并实例化一个 path 对象
path.addCircle(150, 200, 60, Path.Direction.CW);   //在 path 对象中添加一个圆形路径
```

要创建一个折线，可以使用下面的代码。

```
Path mypath=new Path();                  //创建并实例化一个 mypath 对象
mypath.moveTo(50, 100);                  //设置起始点
mypath.lineTo(100, 45);                  //设置第一段直线的结束点
mypath.lineTo(150, 100);                 //设置第二段直线的结束点
mypath.lineTo(200, 80);                  //设置第三段直线的结束点
```

将该路径绘制到画布上的效果如图 14.6 所示。

要创建一个三角形路径，可以使用下面的代码。

```
Path path=new Path();                    //创建并实例化一个 path 对象
path.moveTo(50,50);                      //设置起始点
path.lineTo(100, 10);                    //设置第一条边的结束点，也是第二条边的起始点
path.lineTo(150, 50);                    //设置第二条边的结束点，也是第三条边的起始点
path.close();                            //闭合路径
```

将该路径绘制到画布上的效果如图 14.7 所示。

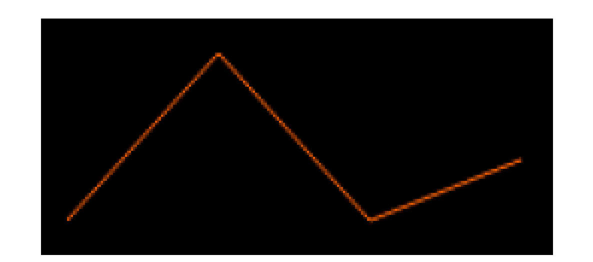


图 14.6　绘制 3 条线组成的折线

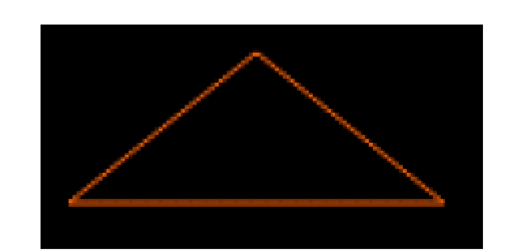


图 14.7　绘制一个三角形

说明：

在创建这个三角形的路径时，如果不使用 close()方法闭合路径，那么绘制的将不是一个三角形，而是一个折线，如图 14.8 所示。

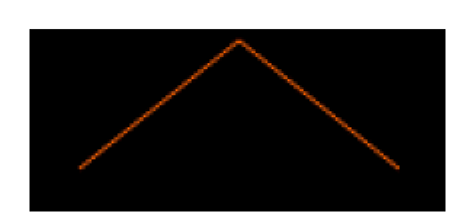
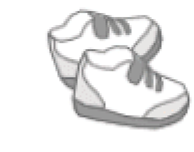
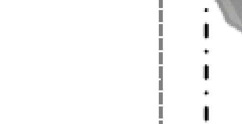


图 14.8　绘制两条线组成的折线

2．将定义好的路径绘制在画布上

使用 Canvas 类提供的 drawPath()方法可以将定义好的路径绘制在画布上。



Note

说明：

在 Android 的 Canvas 类中，还提供了另一个应用路径的方法 drawTextOnPath()，也就是沿着指定的路径绘制字符串，使用该方法可绘制环型文字。

【例 14.5】 在 Eclipse 中创建 Android 项目，实现在屏幕上绘制圆形路径、折线路径、三角形路径，以及绕路径的环形文字。

实例位置：光盘\MR\Instance\14\14.5

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先创建一个画笔，并设置画笔的相关属性，然后创建并绘制一个圆形路径、折线路径和三角形路径，最后再绘制绕路径的环形文字。其具体代码如下：

```
Paint paint=new Paint();                                    //创建一个画笔
paint.setAntiAlias(true);                                   //设置使用抗锯齿功能
paint.setColor(0xFFFF6600);                                 //设置画笔颜色
paint.setTextSize(18);                                      //设置文字大小
paint.setStyle(Style.STROKE);                               //设置填充方式为描边
//绘制圆形路径
Path pathCircle=new Path();                                 //创建并实例化一个 path 对象
pathCircle.addCircle(70, 70, 40, Path.Direction.CCW);//添加逆时针的圆形路径
canvas.drawPath(pathCircle, paint);                         //绘制路径
//绘制折线路径
Path pathLine=new Path();                                   //创建并实例化一个 path 对象
pathLine.moveTo(150, 100);                                  //设置起始点
pathLine.lineTo(200, 45);                                   //设置第一段直线的结束点
pathLine.lineTo(250, 100);                                  //设置第二段直线的结束点
pathLine.lineTo(300, 80);                                   //设置第三段直线的结束点
canvas.drawPath(pathLine, paint);                           //绘制路径
//绘制三角形路径
Path pathTr=new Path();                                     //创建并实例化一个 path 对象
pathTr.moveTo(350,80);                                      //设置起始点
pathTr.lineTo(400, 30);                                     //设置第一条边的结束点，也是第二条边的起始点
pathTr.lineTo(450, 80);                                     //设置第二条边的结束点，也是第三条边的起始点
pathTr.close();                                             //闭合路径
canvas.drawPath(pathTr, paint);                             //绘制路径
//绘制绕路径的环形文字
```





Note

```
String str="风萧萧兮易水寒，壮士一去兮不复还";
Path path=new Path();                              //创建并实例化一个 path 对象
path.addCircle(550, 100, 48, Path.Direction.CW);   //添加顺时针的圆形路径
paint.setStyle(Style.FILL);                        //设置画笔的填充方式
canvas.drawTextOnPath(str, path,0, -18, paint);    //绘制绕路径文字
```

运行本实例，将显示如图 14.9 所示的运行效果。

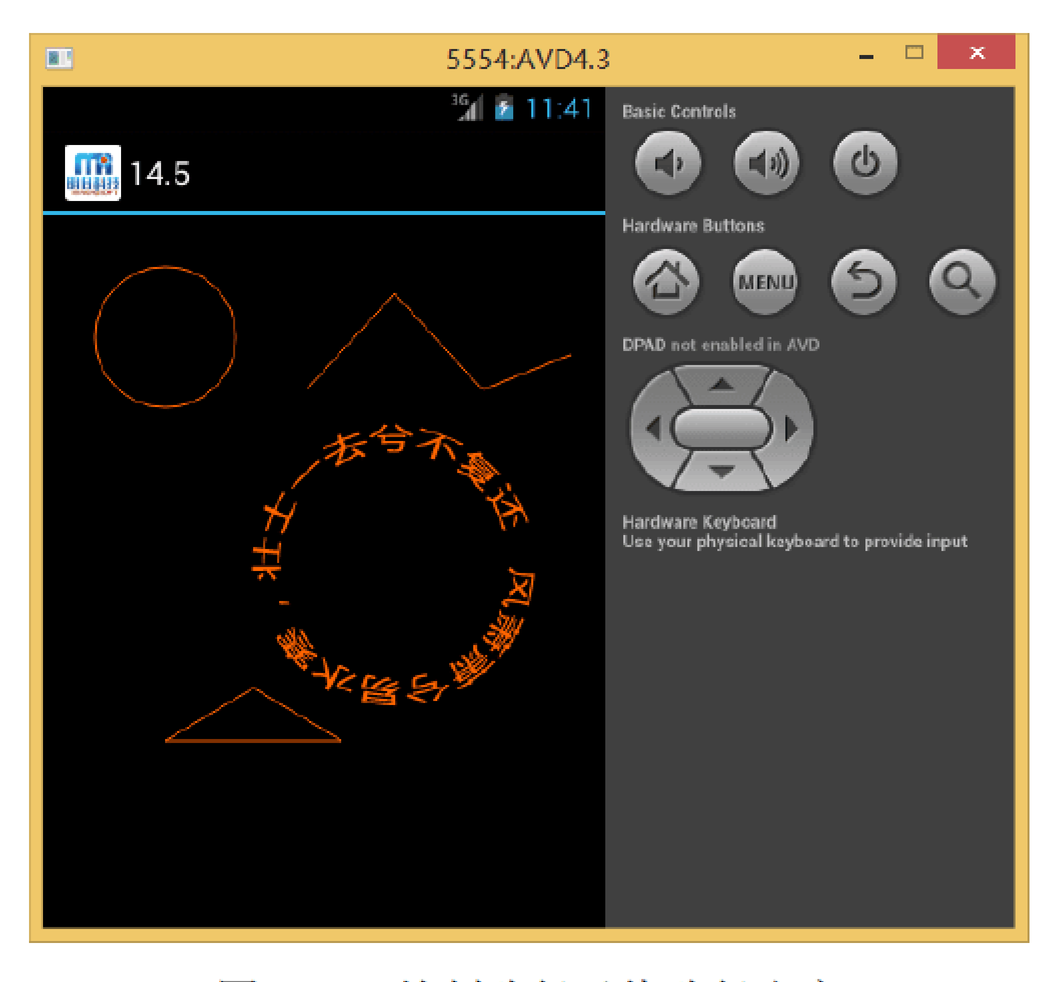


图 14.9　绘制路径及绕路径文字

14.2.4　绘制图片

在 Android 中，Canvas 类不仅可以绘制几何图形、文件和路径，还可用来绘制图片。要想使用 Canvas 类绘制图片，只需要使用 Canvas 类提供的如表 14.6 所示的方法来将 Bitmap 对象中保存的图片绘制到画布上即可。

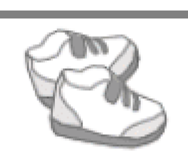
表 14.6　Canvas 类提供的绘制图片的常用方法

方　　法	描　　述
drawBitmap(Bitmap bitmap, Rect src, RectF dst, Paint paint)	用于从指定点绘制从源位图中“挖取”的一块
drawBitmap(Bitmap bitmap, float left, float top, Paint paint)	用于在指定点绘制位图
drawBitmap(Bitmap bitmap, Rect src, Rect dst, Paint paint)	用于从指定点绘制从源位图中“挖取”的一块

例如，从源位图上“挖取”从（0,0）点到（500,300）点的一块图像，然后绘制到画布的（50,50）点到（450,350）点所指区域，可以使用下面的代码。

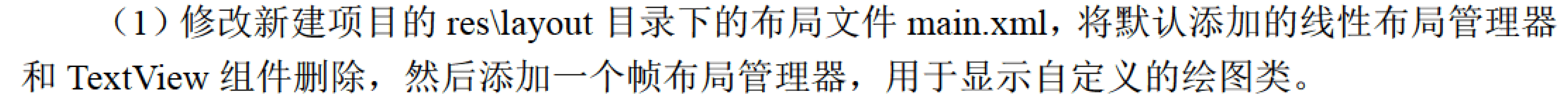
```
Rect src=new Rect(0,0,500,300);            //设置挖取的区域
Rect dst=new Rect(50,50,450,350);          //设置绘制的区域
canvas.drawBitmap(bm, src, dst, paint);    //绘制图片
```

【例 14.6】 在 Eclipse 中创建 Android 项目，实现在屏幕上绘制 Android 的机器人。



 **实例位置：光盘\MR\Instance\14\14.6**

程序的开发步骤如下：


（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 AndroidIco，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先创建一个画笔，并设置画笔的相关属性，然后绘制机器人的头、眼睛、天线、身体、胳膊和腿。其具体代码如下：

```
Paint paint=new Paint();                                  //采用默认设置创建一个画笔
paint.setAntiAlias(true);                                 //使用抗锯齿功能
paint.setColor(0xFFA4C739);                               //设置画笔的颜色为绿色
//绘制机器人的头
RectF rectf_head=new RectF(10, 10, 100, 100);
rectf_head.offset(100, 20);
canvas.drawArc(rectf_head, -10, -160, false, paint);      //绘制弧
//绘制眼睛
paint.setColor(Color.WHITE);                              //设置画笔的颜色为白色
canvas.drawCircle(135, 53, 4, paint);                     //绘制圆
canvas.drawCircle(175, 53, 4, paint);                     //绘制圆
paint.setColor(0xFFA4C739);                               //设置画笔的颜色为绿色
//绘制天线
paint.setStrokeWidth(2);                                  //设置笔触的宽度
canvas.drawLine(120, 15, 135, 35, paint);                 //绘制线
canvas.drawLine(190, 15, 175, 35, paint);                 //绘制线
//绘制身体
canvas.drawRect(110, 75, 200, 150, paint);                //绘制矩形
RectF rectf_body=new RectF(110,140,200,160);
canvas.drawRoundRect(rectf_body, 10, 10, paint);          //绘制圆角矩形
//绘制胳膊
RectF rectf_arm=new RectF(85,75,105,140);
canvas.drawRoundRect(rectf_arm, 10, 10, paint);           //绘制左侧的胳膊
rectf_arm.offset(120, 0);                                 //设置在 X 轴上偏移 120 像素
canvas.drawRoundRect(rectf_arm, 10, 10, paint);           //绘制右侧的胳膊
//绘制腿
RectF rectf_leg=new RectF(125,150,145,200);
canvas.drawRoundRect(rectf_leg, 10, 10, paint);           //绘制左侧的腿
rectf_leg.offset(40, 0);                                  //设置在 X 轴上偏移 40 像素
canvas.drawRoundRect(rectf_leg, 10, 10, paint);           //绘制右侧的腿
```

运行本实例，将显示如图 14.10 所示的运行效果。



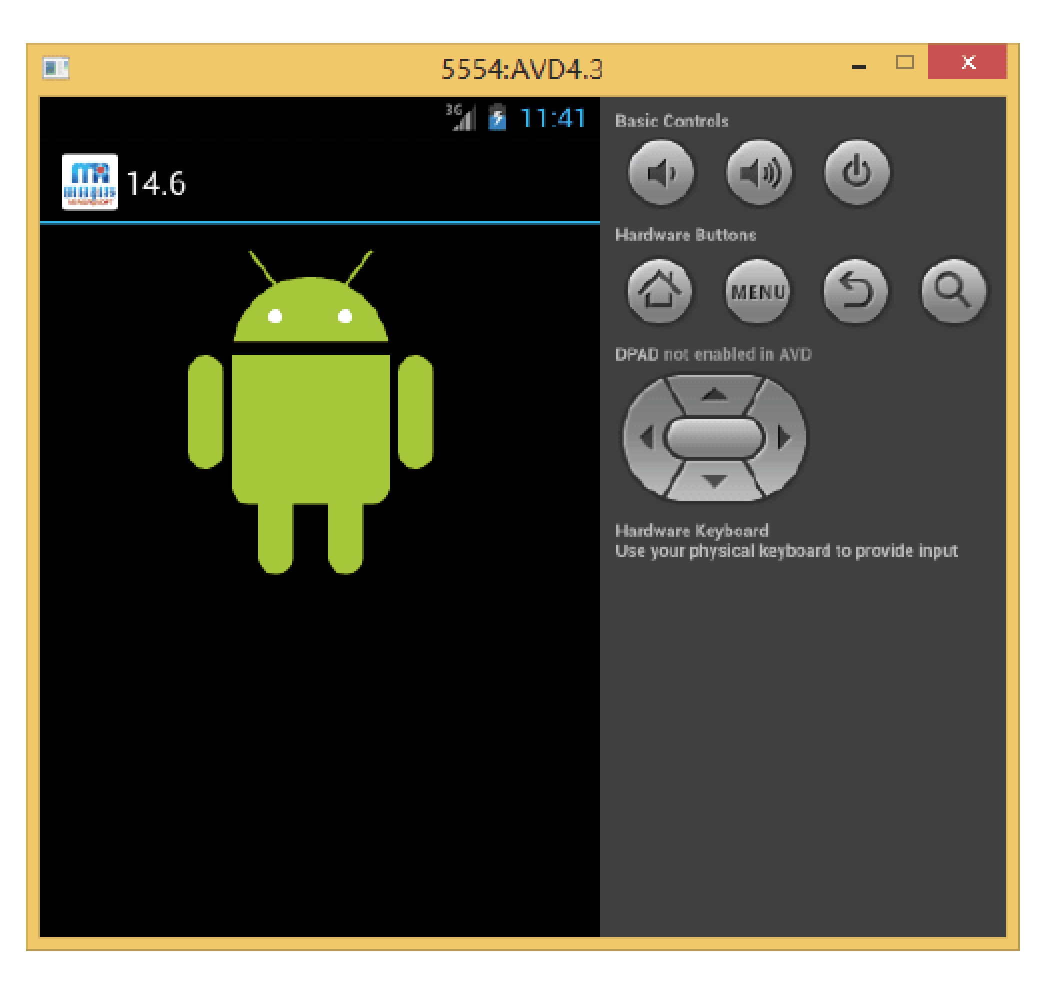




图 14.10　在屏幕上绘制 Android 的机器人

14.3　常见的图像特效

在 Android 中，不仅可以绘制图形，还可以为图形添加特效。例如，对图形进行旋转、缩放、倾斜、扭曲和渲染等。下面将分别介绍如何为图形添加这些特效。

14.3.1　旋转图像

使用 Android 提供的 android.graphics.Matrix 类的 setRotate()、postRotate()和 preRotate()方法可以对图像进行旋转。



说明：

在 Android API 中，提供了 3 种方式，即 setXXX()、postXXX()和 preXXX()方法，其中，setXXX()方法用于直接设置 Matrix 的值，每使用一次 setXXX()方法，整个 Matrix 都会变换；postXXX()方法用于采用后乘的方式为 Matrix 设置值，可以连续多次使用 post 完成多个变换；preXXX()方法用于采用前乘的方式为 Matrix 设置值，使用 preXXX()方法设置的操作最先发生。

由于这 3 个方法除了方法名不同外，其他语法格式均相同。下面将以 setRotate()方法为例来介绍其语法格式。setRotate()方法有以下两种语法格式。

☑　setRotate(float degrees)

使用该语法格式可以控制 Matrix 进行旋转，float 类型的参数用于指定旋转的角度。例如，创建一个 Matrix 的对象，并将它旋转 30 度，可以使用下面的代码。

```
Matrix matrix=new Matrix();                //创建一个 Matrix 的对象
matrix.setRotate(30);                      //将 Matrix 的对象旋转 30 度
```



☑ setRotate(float degrees, float px, float py)

使用该语法格式可以控制 Matrix 以参数 px 和 py 为轴心进行旋转，float 类型的参数用于指定旋转的角度。例如，创建一个 Matrix 的对象，并将它以（10,10）为轴心旋转 30 度，可以使用下面的代码。

Note

```
Matrix matrix=new Matrix();                //创建一个 Matrix 的对象
matrix.setRotate(30,10,10);                //将 Matrix 的对象旋转 30 度
```

创建 Matrix 的对象，并对其进行旋转后，还需要应用该 Matrix 对图像或组件进行控制。在 Canvas 类中提供了一个 drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)方法，可以在绘制图像的同时应用 Matrix 上的变化。例如，需要将一个图像旋转 30 度后，再绘制到画布上可以使用下面的代码。

```
Paint paint=new Paint();
Bitmap bitmap=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.rabbit);
Matrix matrix=new Matrix();
matrix.setRotate(30);
canvas.drawBitmap(bitmap, matrix, paint);
```

【例 14.7】 在 Eclipse 中创建 Android 项目，实现应用 Matrix 旋转图像。

 **实例位置：光盘\MR\Instance\14\14.7**

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并绘制一张背景图像，然后在（0,0）点的位置绘制要旋转图像的原图，再绘制以（0,0）点为轴心旋转 30 度的图像，最后绘制以（87,87）点为轴心旋转 90 度的图像。其具体代码如下：

```
Paint paint=new Paint();                                  //定义一个画笔
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.background);
canvas.drawBitmap(bitmap_bg, 0, 0, paint);                //绘制背景图像
Bitmap bitmap_rabbit=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.rabbit);
canvas.drawBitmap(bitmap_rabbit, 0, 0, paint);            //绘制原图
//应用 setRotate(float degrees)方法旋转图像
Matrix matrix=new Matrix();
matrix.setRotate(30);                                     //以（0,0）点为轴心转换 30 度
canvas.drawBitmap(bitmap_rabbit, matrix, paint);          //绘制图像并应用 matrix 的变换
//应用 setRotate(float degrees, float px, float py)方法旋转图像
Matrix m=new Matrix();
m.setRotate(90,87,87);                                    //以（87,87）点为轴心转换 90 度
canvas.drawBitmap(bitmap_rabbit, m, paint);               //绘制图像并应用 matrix 的变换
```



Note

运行本实例，将显示如图 14.11 所示的运行效果。

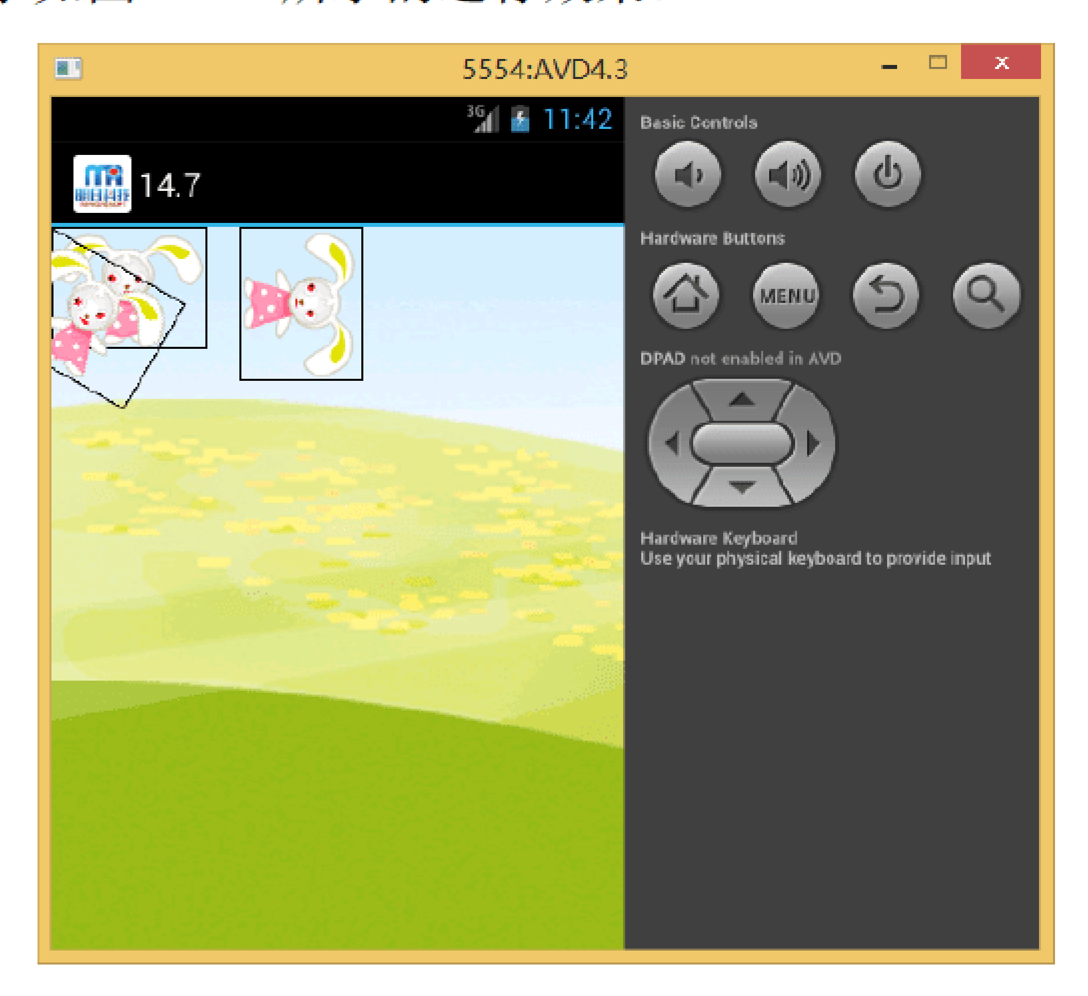



图 14.11　旋转图像

14.3.2　缩放图像

使用 Android 提供的 android.graphics.Matrix 类的 setScale()、postScale()和 preScale()方法可对图像进行缩放。由于这 3 个方法除了方法名不同外，其他语法格式均相同，下面将以 setScale()方法为例来介绍其语法格式。setScale()方法有以下两种语法格式。

☑　setScale(float sx, float sy)

使用该语法格式可以控制 Matrix 进行缩放，参数 sx 和 sy 用于指定 X 轴和 Y 轴的缩放比例。例如，创建一个 Matrix 的对象，并将它在 X 轴上缩放 30%，Y 轴上缩放 20%，可以使用下面的代码。

```
Matrix matrix=new Matrix();                    //创建一个 Matrix 的对象
matrix.setScale(0.3f, 0.2f);                   //缩放 Matrix 对象
```

☑　setScale(float sx, float sy, float px, float py)

使用该语法格式可以控制 Matrix 以参数 px 和 py 为轴心进行缩放，参数 sx 和 sy 用于指定 X 和 Y 轴的缩放比例。例如，创建一个 Matrix 的对象，并将它以（100,100）为轴心，在 X 轴和 Y 轴均缩放 30%，可以使用下面的代码。

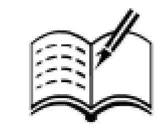
```
Matrix matrix=new Matrix();                    //创建一个 Matrix 的对象
matrix. setScale (30,30,100,100);              //缩放 Matrix 对象
```

创建 Matrix 的对象，并对其进行缩放后，还需要应用该 Matrix 对图像或组件进行控制。同旋转图像一样，也可应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)方法，在绘制图像的同时应用 Matrix 上的变化。下面将通过一个具体的实例来说明如何对图像进行缩放。

【例 14.8】 在 Eclipse 中创建 Android 项目，实现应用 Matrix 缩放图像。



 实例位置：光盘\MR\Instance\14\14.8



程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并绘制一张背景图像，然后以（0,0）点为轴心绘制在 X 轴和 Y 轴上均缩放 200%的图像，再以（156,156）点为轴心绘制在 X 轴和 Y 轴上均缩放 80%的图像，最后在（0,0）点的位置绘制要缩放图像的原图。其具体代码如下：

```
Paint paint=new Paint();                                  //定义一个画笔
paint.setAntiAlias(true);
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.background);
canvas.drawBitmap(bitmap_bg, 0, 0, paint);                //绘制背景
Bitmap bitmap_rabbit=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.rabbit);
//应用 setScale(float sx, float sy)方法缩放图像
Matrix matrix=new Matrix();
matrix.setScale(2f, 2f);                                  //以(0,0)点为轴心将图像在 X 轴和 Y 轴均缩放 200%
canvas.drawBitmap(bitmap_rabbit, matrix, paint);          //绘制图像并应用 matrix 的变换
//应用 setScale(float sx, float sy, float px, float py) 方法缩放图像
Matrix m=new Matrix();
m.setScale(0.8f,0.8f,156,156);                            //以（156,156）点为轴心将图像在 X 轴和 Y 轴均缩放 80%
canvas.drawBitmap(bitmap_rabbit, m, paint);               //绘制图像并应用 matrix 的变换
canvas.drawBitmap(bitmap_rabbit, 0, 0, paint);            //绘制原图
```

运行本实例，将显示如图 14.12 所示的运行效果。

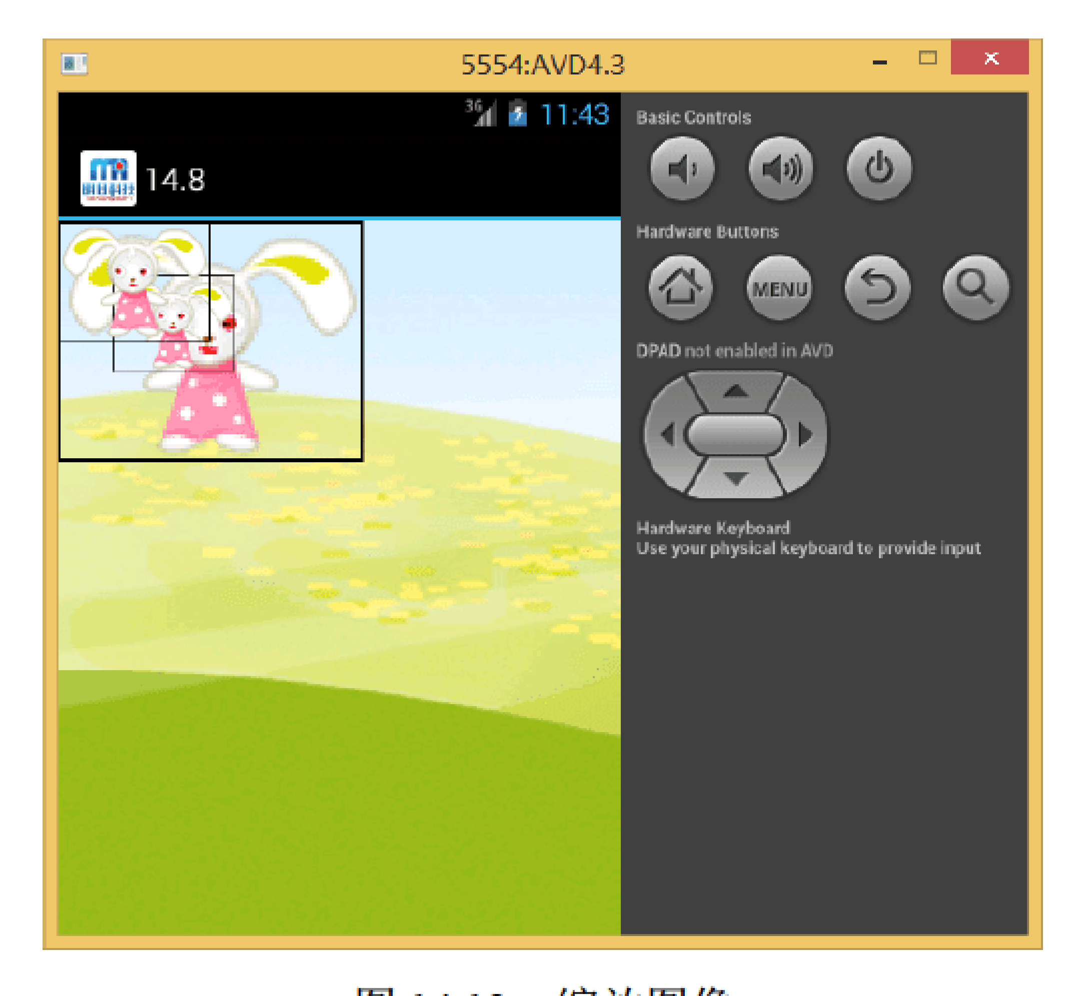
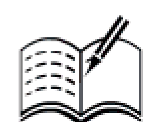


图 14.12　缩放图像



14.3.3　倾斜图像



使用 Android 提供的 android.graphics.Matrix 类的 setSkew()、postSkew()和 preSkew()方法可对图像进行倾斜。由于这 3 个方法除了方法名不同外，其他语法格式均相同，下面将以 setSkew()方法为例来介绍其语法格式。setSkew()方法有以下两种语法格式。

☑　setSkew(float sx, float sy)

使用该语法格式可以控制 Matrix 进行倾斜，参数 sx 和 sy 用于指定 X 轴和 Y 轴的倾斜量。例如，创建一个 Matrix 的对象，并将它在 X 轴上倾斜 0.3，Y 轴上不倾斜，可以使用下面的代码。

```
Matrix matrix=new Matrix();                    //创建一个 Matrix 的对象
matrix.setScale(0.3f, 0);                      //倾斜 Matrix 对象
```

☑　setSkew(float sx, float sy, float px, float py)

使用该语法格式可以控制 Matrix 以参数 px 和 py 为轴心进行倾斜，参数 sx 和 sy 用于指定 X 轴和 Y 轴的倾斜量。例如，创建一个 Matrix 的对象，并将它以（100,100）为轴心，在 X 轴和 Y 轴均倾斜 0.1，可以使用下面的代码。

```
Matrix matrix=new Matrix();                    //创建一个 Matrix 的对象
matrix. setScale (0.1f,0.1f,100,100);          //缩放 Matrix 对象
```

创建 Matrix 的对象，并对其进行倾斜后，还需要应用该 Matrix 对图像或组件进行控制。同旋转图像一样，也可应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)方法，在绘制图像的同时应用 Matrix 上的变化。下面通过一个具体的实例来说明如何对图像进行倾斜。

【例 14.9】 在 Eclipse 中创建 Android 项目，实现应用 Matrix 倾斜图像。

实例位置：光盘\MR\Instance\14\14.9

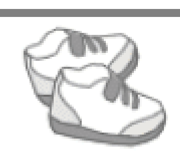
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并绘制一张背景图像，然后以（0,0）点为轴心绘制在 X 轴上倾斜 2，在 Y 轴上倾斜 1 的图像，再以（78,69）点为轴心绘制在 X 轴上倾斜-0.5 的图像，最后在（0,0）点的位置绘制要缩放图像的原图。其具体代码如下：

```
Paint paint=new Paint();                                  //定义一个画笔
paint.setAntiAlias(true);
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.background);
```





```
canvas.drawBitmap(bitmap_bg, 0, 0, paint);              //绘制背景
Bitmap bitmap_rabbit=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.rabbit);
//应用 setSkew(float sx, float sy)方法倾斜图像
Matrix matrix=new Matrix();
matrix.setSkew(2f, 1f);                         //以（0,0）点为轴心将图像在 X 轴上倾斜 2，在 Y 轴上倾斜 1
canvas.drawBitmap(bitmap_rabbit, matrix, paint);        //绘制图像并应用 matrix 的变换
//应用 setSkew(float sx, float sy, float px, float py) 方法倾斜图像
Matrix m=new Matrix();
m.setSkew(-0.5f, 0f,78,69);                             //以（78,69）点为轴心将图像在 X 轴上倾斜-0.5
canvas.drawBitmap(bitmap_rabbit, m, paint);             //绘制图像并应用 matrix 的变换
canvas.drawBitmap(bitmap_rabbit, 0, 0, paint);          //绘制原图
```

运行本实例，将显示如图 14.13 所示的运行效果。

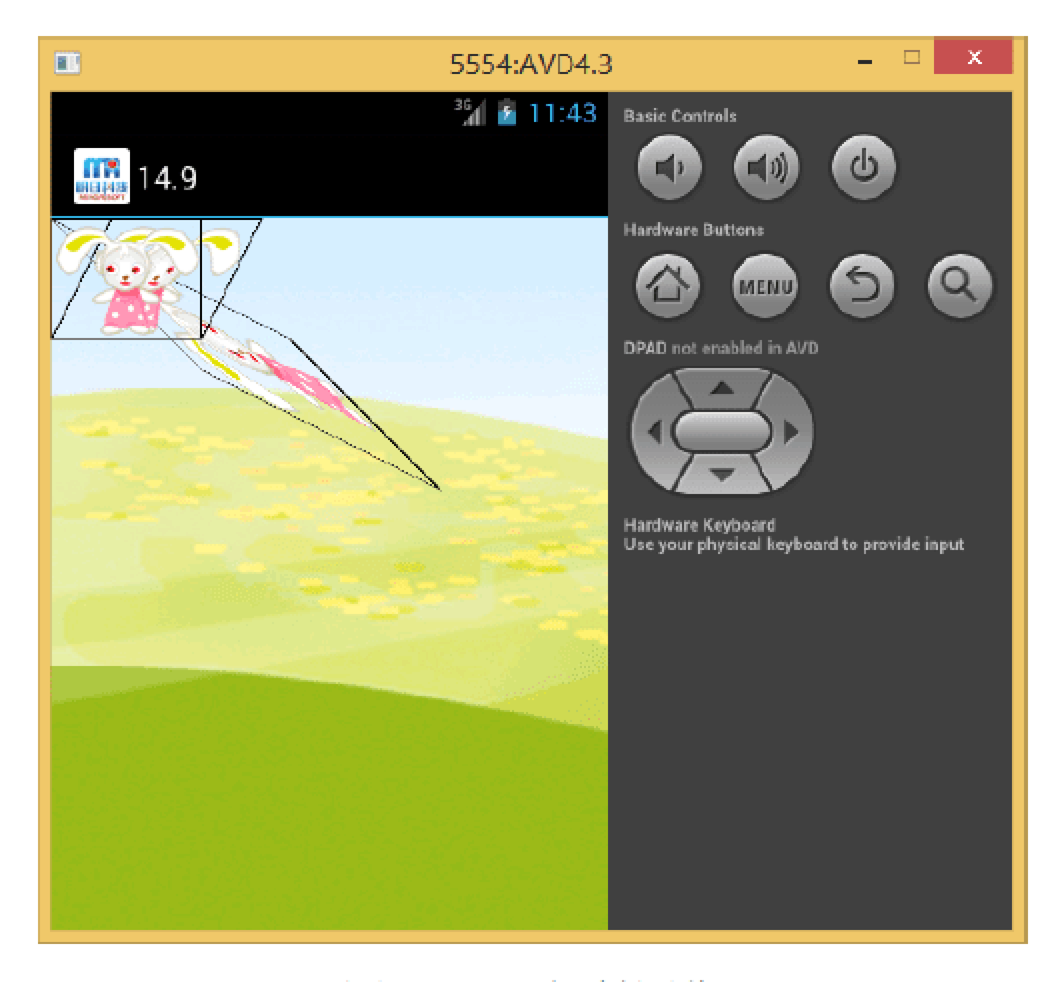


图 14.13 倾斜图像

14.3.4 平移图像

使用 Android 提供的 android.graphics.Matrix 类的 setTranslate()、postTranslate()和 preTranslate()方法可对图像进行平移。由于这 3 个方法除了方法名不同外，其他语法格式均相同，下面将以 setTranslate()方法为例来介绍其语法格式。setTranslate()方法的语法格式如下：

```
setTranslate(float dx, float dy)
```

在该语法中，参数 dx 和 dy 用于指定将 Matrix 移动到的位置的 X 和 Y 坐标。

例如，创建一个 Matrix 的对象，并将它平移到（100,100）的位置，可以使用下面的代码。

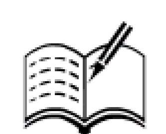
```
Matrix matrix=new Matrix();                             //创建一个 Matrix 的对象
matrix.setTranslate(100,50);                            //将 matrix 平移到（100,50）的位置
```

创建 Matrix 的对象，并对其进行平移后，还需要应用该 Matrix 对图像或组件进行控制。同旋转图像一样，也可应用 Canvas 类中提供的 drawBitmap(Bitmap bitmap, Matrix matrix, Paint paint)



方法，在绘制图像的同时应用 Matrix 上的变化。下面将通过一个具体的实例来说明如何对图像进行倾斜。

【例 14.10】 在 Eclipse 中创建 Android 项目，实现应用 Matrix 将图像旋转后再平移。



实例位置：光盘\MR\Instance\14\14.10

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中，获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并绘制一张背景图像，然后在（0,0）点的位置绘制要缩放图像的原图，再创建一个 Matrix 的对象，并将其旋转 30 度后，再将其平移到指定位置，最后绘制应用 Matrix 变换的图像。其具体代码如下：

```
Paint paint=new Paint();                                    //定义一个画笔
paint.setAntiAlias(true);                                   //使用抗锯齿功能
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.background);
canvas.drawBitmap(bitmap_bg, 0, 0, paint);                  //绘制背景
Bitmap bitmap_rabbit=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.rabbit);
canvas.drawBitmap(bitmap_rabbit, 0, 0, paint);              //绘制原图
Matrix matrix=new Matrix();                                 //创建一个 Matrix 的对象
matrix.setRotate(30);                                       //将 matrix 旋转 30 度
matrix.postTranslate(100,50);                               //将 matrix 平移到（100,50）的位置
canvas.drawBitmap(bitmap_rabbit, matrix, paint);            //绘制图像并应用 matrix 的变换
```

运行本实例，将显示如图 14.14 所示的运行效果。

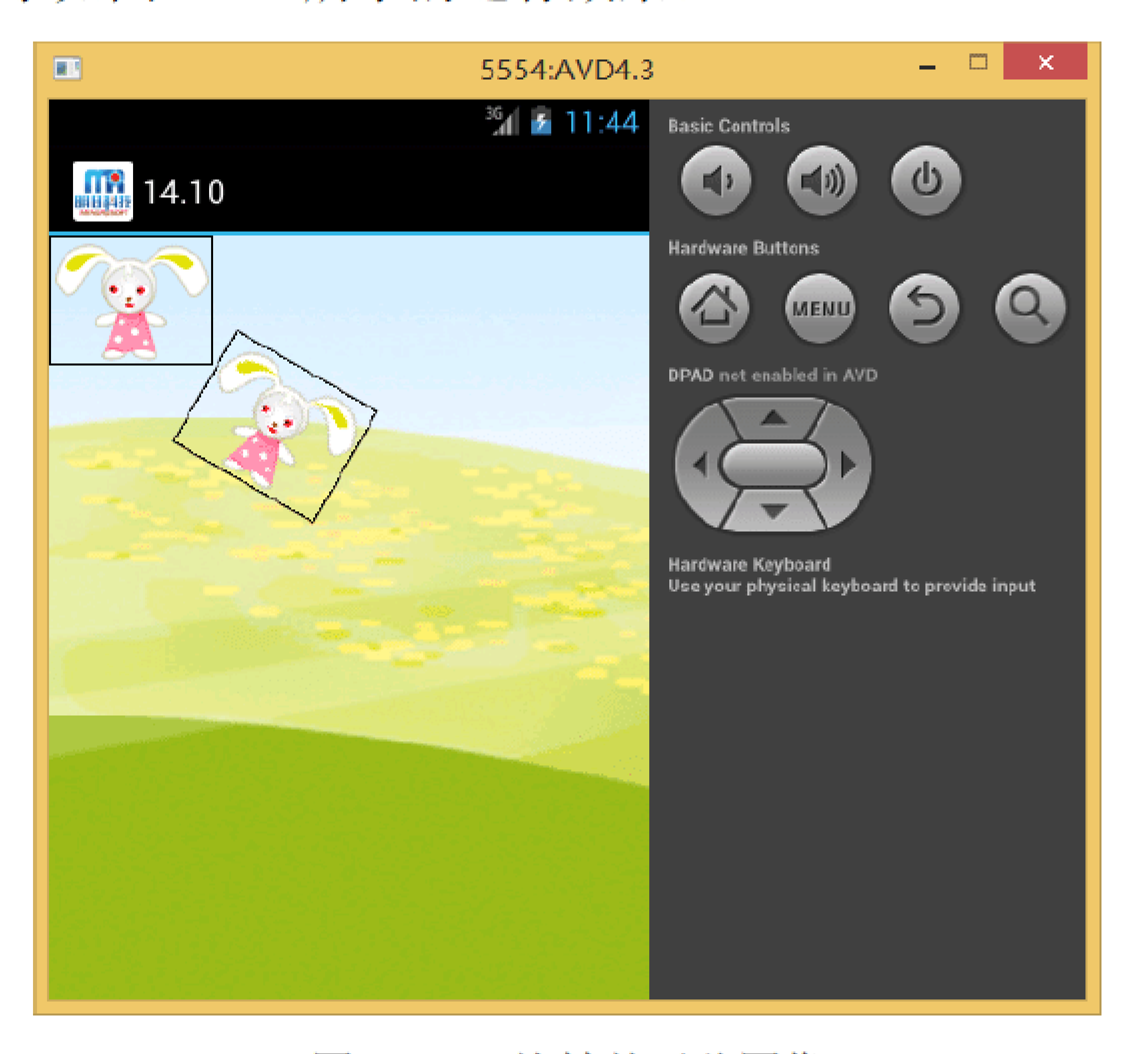
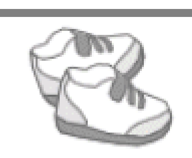
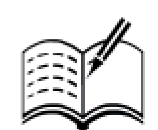


图 14.14　旋转并平移图像





14.3.5 使用 BitmapShader 渲染图像



Note

在 Android 中，提供的 BitmapShader 类主要用来渲染图像。如果需要将一张图片裁剪成椭圆或圆形等形状显示到屏幕上时，就可以使用 BitmapShader 类来实现。使用 BitmapShader 来渲染图像的基本步骤如下：

（1）创建 BitmapShader 类的对象，可以通过以下的构造方法进行创建。

```
BitmapShader(Bitmap bitmap, Shader.TileMode tileX, Shader.TileMode tileY)
```

其中，参数 bitmap 用于指定一个位图对象，通常是要用来渲染的原图像；参数 tileX 用于指定在水平方向上图像的重复方式；参数 tileY 用于指定在垂直方向上图像的重复方式。例如，要创建一个在水平方向上重复、在垂直方向上镜像的 BitmapShader 对象可以使用下面的代码。

```
BitmapShader bitmapshader= new BitmapShader(bitmap_bg,TileMode.REPEAT,TileMode.MIRROR);
```



说明：

Shader.TileMode 类型的参数包括 CLAMP、MIRROR 和 REPEAT 3 个可选值，其中，CLAMP 为使用边界颜色来填充剩余的空间；MIRROR 为采用镜像方式；REPEAT 为采用复复方式。

（2）通过 Paint 的 setShader()方法来设置渲染对象。

（3）在绘制图像时，使用已经设置了 setShader()方法的画笔。

下面通过一个具体的实例来说明如何使用 BitmapShader 渲染图像。

【例 14.11】 在 Eclipse 中创建 Android 项目，应用 BitmapShader 实现平铺的画布背景和椭圆形的图片。

实例位置：光盘\MR\Instance\14\14.11

程序的开发步骤如下：

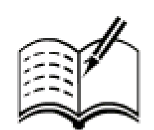
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并设置其使用抗锯齿功能，然后应用 BitmapShader 实现平铺的画布背景，这里使用的是一张机器人图片，接下来再绘制一张椭圆形的图片。其具体代码如下：

```
Paint paint=new Paint();                                        //定义一个画笔
paint.setAntiAlias(true);                                       //使用抗锯齿功能
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.android);
```






```
//创建一个在水平和垂直方向都重复的 BitmapShader 对象
BitmapShader bitmapshader= new BitmapShader(bitmap_bg,TileMode.REPEAT,TileMode.REPEAT);
paint.setShader(bitmapshader);                //设置渲染对象
canvas.drawRect(0, 0, view_width, view_height, paint);//绘制使用 BitmapShader 渲染的矩形
Bitmap bm=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.img02);
//创建一个在水平方向上重复，在垂直方向上镜像的 BitmapShader 对象
BitmapShader bs= new BitmapShader(bm,TileMode.REPEAT,TileMode.MIRROR);
paint.setShader(bs);                          //设置渲染对象
RectF oval=new RectF(0,0,280,180);
canvas.translate(40, 20);                     //将画面在 X 轴上平移 40 像素，在 Y 轴上平移 20 像素
canvas.drawOval(oval, paint);                 //绘制一个使用 BitmapShader 渲染的椭圆形
```

运行本实例，将显示如图 14.15 的所示的运行效果。

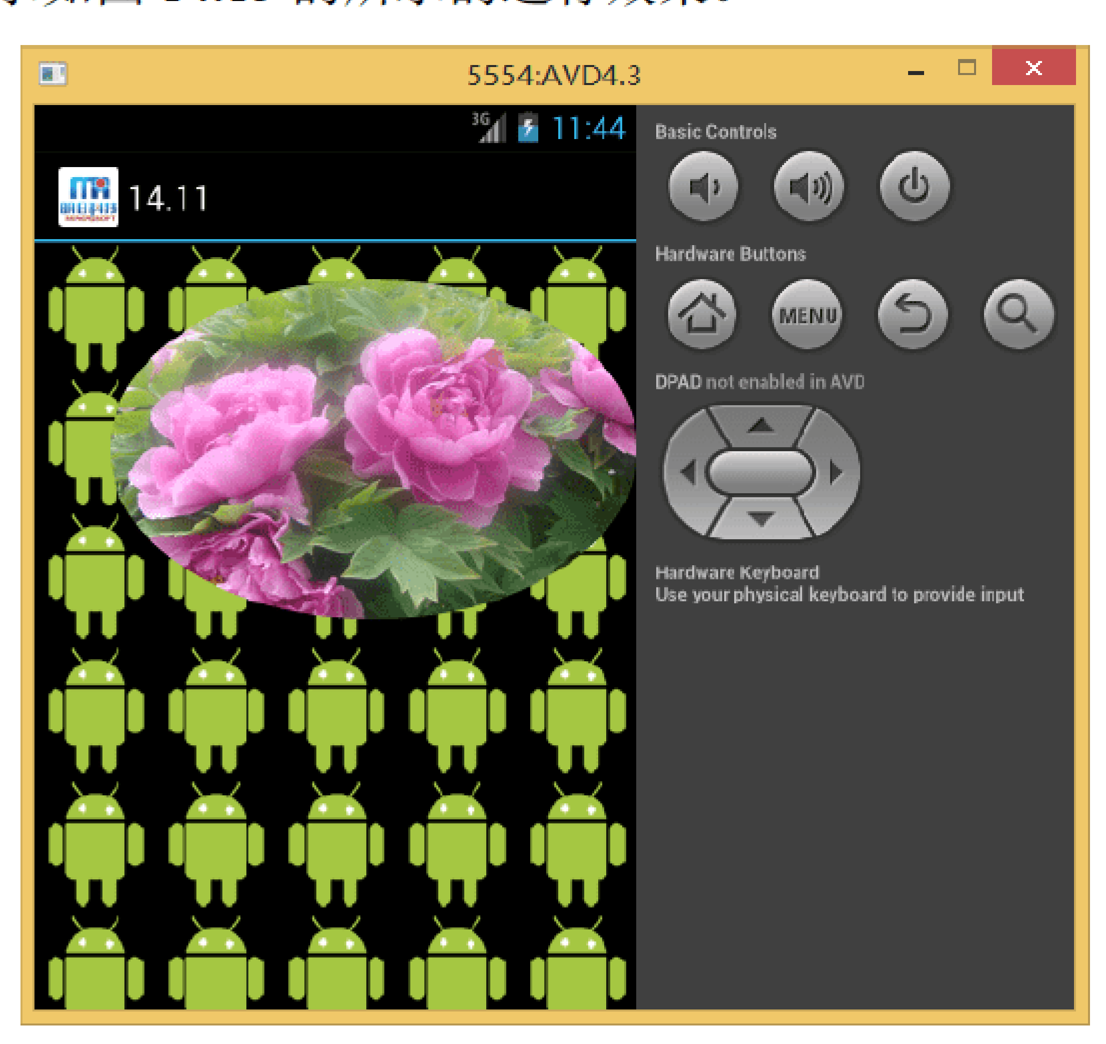


图 14.15　显示平铺背景和椭圆形的图片

14.4　Android 中的动画

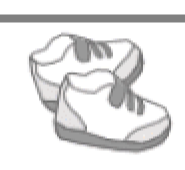
在应用 Android 进行项目开发时，经常需要涉及动画，特别是在进行游戏开发时。Android 中的动画通常可以分为逐帧动画和补间动画两种。下面将分别介绍如何实现这两种动画。

14.4.1　实现逐帧动画

逐帧动画和就是顺序播放事先准备好的静态图像，利用人眼的“视觉暂留”的原理，给用户造成动画的错觉。实现逐帧动画比较简单，只需要经过以下两个步骤就可以实现。

（1）在 Android XML 资源文件中定义一组用于生成动画的图片资源。

要在 Android XML 资源文件中定义一组生成动画的图片资源，可以使用包含一系列

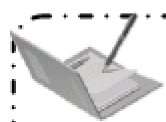


<item></item>子标记的<animation-list></animation-list>标记来实现。其具体语法格式如下：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
android:oneshot="true|false">
    <item android:drawable="@drawable/图片资源名 1" android:duration="integer" />
    …     <!-- 省略了部分<item></item>标记 -->
    <item android:drawable="@drawable/图片资源名 n" android:duration="integer" />
</animation-list>
```

在上面的语法中，android:oneshot 属性用于设置是否循环播放，默认值为 true，也就是循环播放；android:drawable 属性用于指定要显示的图片资源；android:duration 属性用于指定图片资源持续的时间。

（2）使用步骤（1）中定义的动画资源，通常情况下，可以将其作为组件的背景使用。例如，可以在布局文件中添加一个线性布局管理器，然后将该布局管理器的 android:background 属性设置为定义的动画资源，也可以将定义的动画资源作为 ImageView 的背景使用。



说明：

在 Android 中还支持在 Java 代码中创建逐帧动画。具体的步骤是：首先创建 AnimationDrawable 对象，然后调用 addFrame()方法向动画中添加帧，每调用一个 addFrame()方法，将添加一个帧。

14.4.2 实现补间动画

补间动画就是通过对场景里的对象不断进行图像变化来产生动画效果。在实现补间动画时，只需要定义动画开始和结束的“关键帧”，其他过渡帧由系统自动计算并补齐。在 Android 中，提供了以下 4 种补间动画。

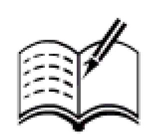
1. 透明度渐变动画（AlphaAnimation）

透明度渐变动画就是指通过 View 组件透明度的变化来实现 View 的渐隐渐显的效果。它主要通过为动画指定开始时的透明度和结束时的透明度，以及持续时间来创建动画。同逐帧动画一样，也可以在 XML 文件中定义透明度渐变动画的动画资源文件。其基本语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@[package:]anim/interpolator_resource">
<alpha
      android:repeatMode="reverse|restart"
      android:repeatCount="次数|infinite"
      android:duration="Integer"
      android:fromAlpha="float"
      android:toAlpha="float" />
</set>
```

在上面的语法中，各属性说明如表 14.7 所示。





Note

表 14.7 定义透明度渐变动画时常用的属性

属 性	描 述
android:interpolator	用于控制动画的变化速度，使动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值如表 14.8 所示
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒
android:fromAlpha	用于指定动画开始时的透明度，值为 0.0 代表完全透明，值为 1.0 代表完全不透明
android:toAlpha	用于指定动画结束时的透明度，值为 0.0 代表完全透明，值为 1.0 代表完全不透明


表 14.8 android:interpolator 属性的常用属性值

属 性 值	描 述
@android:anim/linear_interpolator	动画一直在做匀速改变
@android:anim/accelerate_interpolator	动画在开始的地方改变较慢，然后开始加速
@android:anim/decelerate_interpolator	在动画开始的地方改变速度较快，然后开始减速
@android:anim/accelerate_decelerate_interpolator	动画在开始和结束的地方改变速度较慢，在中间的时候加速
@android:anim/cycle_interpolator	动画循环播放特定的次数，变化速度按正弦曲线改变
@android:anim/bounce_interpolator	动画结束的地方采用弹球效果
@android:anim/anticipate_overshoot_interpolator	在动画开始的地方先向后退一小步，再开始动画，到结束的地方再超出一小步，最后回到动画结束的地方
@android:anim/overshoot_interpolator	动画快速到达终点，并超出一小步最后回到动画结束的地方
@android:anim/anticipate_interpolator	在动画开始的地方先向后退一小步，再快速到达动画结束的地方

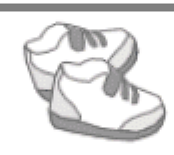
例如，使用下面的代码可以定义一个让 View 组件从完全透明到完全不透明，持续时间为 2 秒钟的动画。

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="0"
        android:toAlpha="1"
        android:duration="2000"/>
</set>
```

2. 旋转动画（RotateAnimation）

旋转动画就是通过为动画指定开始时的旋转角度、结束时的旋转角度，以及持续时间来创建动画。在旋转时还可以通过指定轴心点坐标来改变旋转的中心。同透明度渐变动画一样，也可以在 XML 文件中定义旋转动画资源文件。其基本语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@[package:]anim/interpolator_resource">
<rotate
    android:fromDegrees="float"
```

```
        android:toDegrees="float"
        android:pivotX="float"
        android:pivotY="float"
            android:repeatMode="reverse|restart"
            android:repeatCount="次数|infinite"
            android:duration="Integer"/>
</set>
```

Note

在上面的语法中，各属性说明如表 14.9 所示。

表 14.9　定义旋转动画时常用的属性

属　　性	描　　述
android:interpolator	用于控制动画的变化速度，使动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值如表 14.8 所示
android:fromDegrees	用于指定动画开始时旋转的角度
android:toDegrees	用于指定动画结束时旋转的角度
android:pivotX	用于指定轴心点 X 轴的坐标
android:pivotY	用于指定轴心点 Y 轴的坐标
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

例如，使用下面的代码可以定义一个让图片从 0 度转到 360 度、持续时间为 2 秒钟、中心点在图片的中心的动画。

```
<rotate
    android:fromDegrees="0"
    android:toDegrees="360"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000">
</rotate>
```

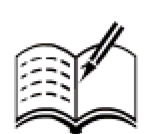
3．缩放动画（ScaleAnimation）

缩放动画就是通过为动画指定开始时的缩放系数、结束时的缩放系数，以及持续时间来创建动画。在缩放时还可以通过指定轴心点坐标来改变缩放的中心。同透明度渐变动画一样，也可以在 XML 文件中定义缩放动画资源文件。其基本语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@[package:]anim/interpolator_resource">
<scale
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
```



```
        android:pivotY="float"
            android:repeatMode="reverse|restart"
            android:repeatCount="次数|infinite"
            android:duration="Integer"/>
</set>
```



在上面的语法中，各属性说明如表 14.10 所示。

表 14.10　定义缩放动画时常用的属性

属　性	描　述
android:interpolator	用于控制动画的变化速度，使动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值如表 14.8 所示
android:fromXScale	用于指定动画开始时水平方向上的缩放系数，值为 1.0 表示不变化
android:toXScale	用于指定动画结束时水平方向上的缩放系数，值为 1.0 表示不变化
android:fromYScale	用于指定动画开始时垂直方向上的缩放系数，值为 1.0 表示不变化
android:toYScale	用于指定动画结束时水平方向上的缩放系数，值为 1.0 表示不变化
android:pivotX	用于指定轴心点 X 轴的坐标
android:pivotY	用于指定轴心点 Y 轴的坐标
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

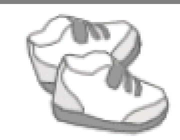
例如，使用下面的代码可以定义一个以图片的中心为轴心点，将图片放大 2 倍、持续时间为 2 秒钟的动画。

```
<scale android:fromXScale="1"
    android:fromYScale="1"
    android:toXScale="2.0"
    android:toYScale="2.0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="2000"/>
```

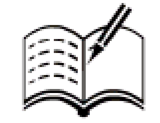
4．平移动画（TranslateAnimation）

平移动画就是通过为动画指定开始时的位置、结束时的位置，以及持续时间来创建动画。同透明度渐变动画一样，也可以在 XML 文件中定义平移动画资源文件。其基本语法格式如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@[package:]anim/interpolator_resource">
<translate
        android:fromXDelta="float"
        android:toXDelta="float"
        android:fromYDelta="float"
        android:toYDelta="float"
            android:repeatMode="reverse|restart"
```

```
        android:repeatCount="次数|infinite"
        android:duration="Integer"/>
</set>
```



在上面的语法中，各属性说明如表 14.11 所示。

表 14.11　定义平移动画时常用的属性

属　　性	描　　述
android:interpolator	用于控制动画的变化速度，使动画效果可以匀速、加速、减速或抛物线速度等各种速度变化，其属性值如表 14.8 所示
android:fromXDelta	用于指定动画开始时水平方向上的起始位置
android:toXDelta	用于指定动画结束时水平方向上的起始位置
android:fromYDelta	用于指定动画开始时垂直方向上的起始位置
android:toYDelta	用于指定动画结束时垂直方向上的起始位置
android:repeatMode	用于设置动画的重复方式，可选值为 reverse（反向）或 restart（重新开始）
android:repeatCount	用于设置动画的重复次数，属性可以是代表次数的数值，也可以是 infinite（无限循环）
android:duration	用于指定动画持续的时间，单位为毫秒

例如，使用下面的代码可以定义一个让图片从（0,0）点到（300,300）点、持续时间为 2 秒钟的动画。

```
<translate
    android:fromXDelta="0"
    android:toXDelta="300"
    android:fromYDelta="0"
    android:toYDelta="300"
    android:duration="2000">
</translate>
```

【例 14.12】 在 Eclipse 中创建 Android 项目，实现旋转、平移、缩放和透明度渐变的补间动画。

实例位置：光盘\MR\Instance\14\14.12

程序的开发步骤如下：

（1）在新建项目的 res 目录中，创建一个名称为 anim 的目录，并在该目录中创建实现旋转、平移、缩放和透明度渐变的动画资源文件。

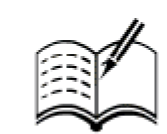
创建名称为 anim_alpha.xml 的 XML 资源文件，在该文件中定义一个实现透明度渐变的动画，该动画为从完全不透明到完全透明，再到完全不透明的渐变过程。其具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha android:fromAlpha="1"
        android:toAlpha="0"
        android:fillAfter="true"
        android:repeatMode="reverse"
        android:repeatCount="1"
```



```
            android:duration="2000"/>
</set>
```

创建名称为 anim_rotate.xml 的 XML 资源文件，在该文件中定义一个实现旋转的动画，该动画为从 0 度旋转到 720 度，再从 360 度旋转到 0 度。其具体代码如下：



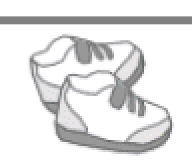
```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromDegrees="0"
        android:toDegrees="720"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000">
    </rotate>
    <rotate
        android:interpolator="@android:anim/accelerate_interpolator"
        android:startOffset="2000"
        android:fromDegrees="360"
        android:toDegrees="0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="2000">
    </rotate>
</set>
```

创建名称为 anim_scale.xml 的 XML 资源文件，在该文件中定义一个实现缩放的动画，该动画首先将原图像放大 2 倍，再逐渐收缩为图像的原尺寸。其具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set   xmlns:android="http://schemas.android.com/apk/res/android">
    <scale android:fromXScale="1"
        android:interpolator="@android:anim/decelerate_interpolator"
        android:fromYScale="1"
        android:toXScale="2.0"
        android:toYScale="2.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="true"
        android:repeatCount="1"
        android:repeatMode="reverse"
        android:duration="2000"/>
</set>
```

创建名称为 anim_translate.xml 的 XML 资源文件，在该文件中定义一个实现平移的动画，该动画为从屏幕的左侧移动到屏幕的右侧，再从屏幕的右侧返回到左侧。其具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
```



Note

```
<translate
        android:fromXDelta="0"
    android:toXDelta="860"
        android:fromYDelta="0"
        android:toYDelta="0"
        android:fillAfter="true"
    android:repeatMode="reverse"
        android:repeatCount="1"
        android:duration="2000">
</translate>
</set>
```

（2）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中再添加一个水平线性布局管理器和一个 ImageView 组件，再向这个水平线性布局管理器中添加 4 个 Button 按钮，最后再设置 ImageView 组件的的左边距和要显示的图片。

（3）打开默认创建的 MainActivity，在 onCreate()方法中，首先获取动画资源文件中创建的动画资源，然后获取要应用动画效果的 ImageView，再获取“旋转”按钮，并为该按钮添加单击事件监听器，在重写的 onClick()方法中，播放“旋转”动画。其具体代码如下：

```
//获取“旋转”动画资源
final Animation rotate=AnimationUtils.loadAnimation(this, R.anim.anim_rotate);
//获取“平移”动画资源
final Animation translate=AnimationUtils.loadAnimation(this, R.anim.anim_translate);
//获取“缩放”动画资源
final Animation scale=AnimationUtils.loadAnimation(this, R.anim.anim_scale);
//获取“透明度变化”动画资源
final Animation alpha=AnimationUtils.loadAnimation(this, R.anim.anim_alpha);
final ImageView iv=(ImageView)findViewById(R.id.imageView1);      //获取要应用动画效果的 ImageView
Button button1=(Button)findViewById(R.id.button1);                //获取“旋转”按钮
button1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        iv.startAnimation(rotate);                                //播放“旋转”动画
    }
});
```

获取“平移”按钮，并为该按钮添加单击事件监听器，在重写的 onClick()方法中，播放“平移”动画。其关键代码如下：

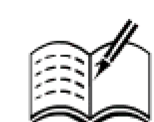
```
iv.startAnimation(translate);                                     //播放“平移”动画
```

获取“缩放”按钮，并为该按钮添加单击事件监听器，在重写的 onClick()方法中，播放“缩放”动画。其关键代码如下：

```
iv.startAnimation(scale);                                         //播放“缩放”动画
```

获取“透明度渐变”按钮，并为该按钮添加单击事件监听器，在重写的 onClick()方法中，





播放“透明度渐变”动画。其关键代码如下：

```
iv.startAnimation(alpha);                                   //播放“透明度渐变”动画
```

运行本实例，单击“旋转”按钮，屏幕中的小猫将旋转，如图 14.16 所示；单击“平移”按钮，屏幕中的小猫将从屏幕的左侧移动到右侧，再从右侧返回左侧；单击“缩放”按钮，屏幕中的小猫将放大 2 倍，再恢复为原来的大小；单击“透明度渐变”按钮，屏幕中的小猫将逐渐隐藏，再逐渐显示。

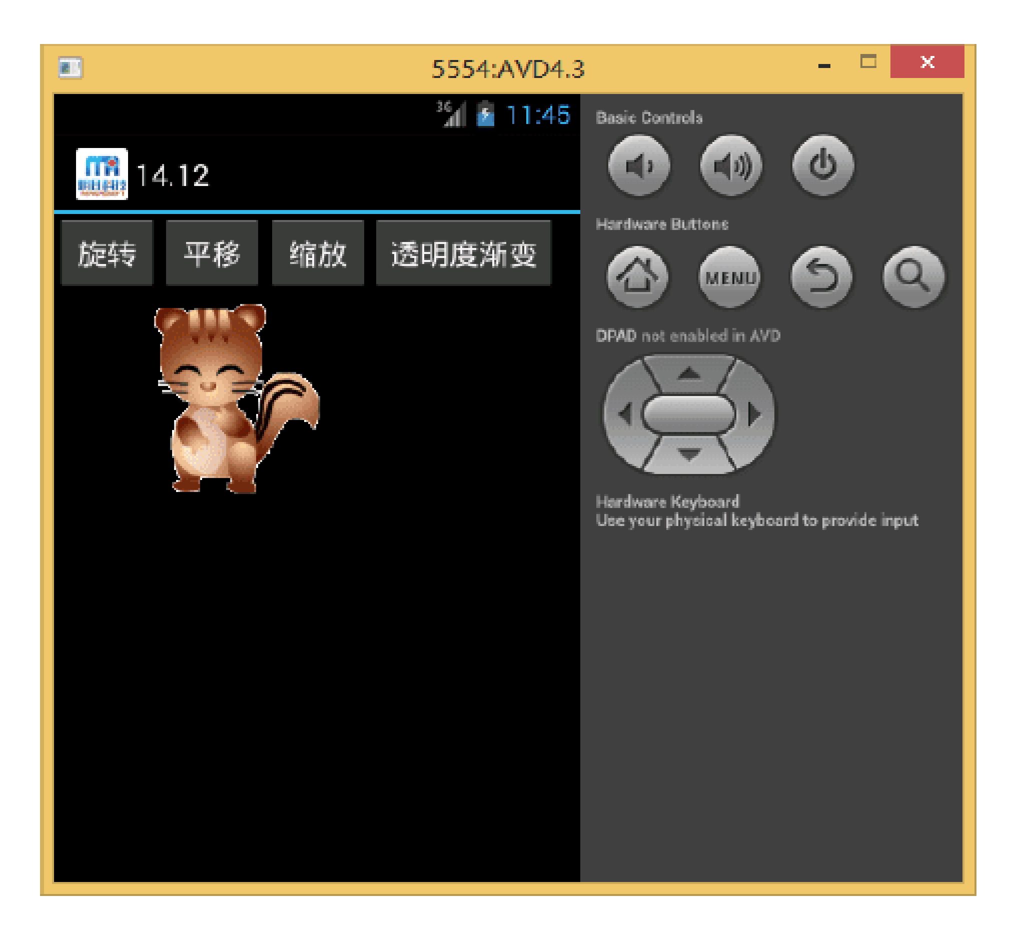


图 14.16　旋转、平移、缩放和透明度渐变的补间动画

14.5　综 合 应 用

14.5.1　实现带描边的圆角图片

【例 14.13】 在 Eclipse 中创建 Android 项目，实现带描边的圆角图片，运行程序，将显示如图 14.17 所示的运行效果。



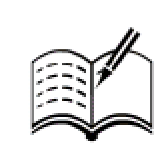
图 14.17　带描边的圆角图片

实例位置：光盘\MR\Instance\14\14.13

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器





Note

和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。

（3）在 MyView 的 onDraw()方法中，首先定义一个画笔，并绘制一张背景图像，然后定义一个要绘制的圆角矩形的区域，并将画布在 X 轴上平移 40 像素，在 Y 轴上平移 20 像素，再绘制一个黑色的两个像素的圆角矩形，作为图片的描边，最后绘制一个使用 BitmapShader 渲染的圆角矩形图片。其具体代码如下：

```
Paint paint=new Paint();                          //定义一个画笔
paint.setAntiAlias(true);                         //使用抗锯齿功能
Bitmap bitmap_bg=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.background);
canvas.drawBitmap(bitmap_bg, 0, 0, paint);        //绘制背景
RectF rect=new RectF(0,0,280,180);
canvas.translate(40, 20);                         //将画布在 X 轴上平移 40 像素，在 Y 轴上平移 20 像素
//为图片添加描边
paint.setStyle(Style.STROKE);                     //设置填充样式为描边
paint.setColor(Color.BLACK);                      //设置颜色为黑色
paint.setStrokeWidth(2);                          //设置笔触宽度为 2 像素
canvas.drawRoundRect(rect, 10, 10, paint);        //绘制一个描边的圆角矩形
paint.setStyle(Style.FILL);                       //设置填充样式为填充
Bitmap bm=BitmapFactory.decodeResource(MainActivity.this.getResources(), R.drawable.img02);
//创建一个在水平方向上重复，在垂直方向上镜像的 BitmapShader 对象
BitmapShader bs= new BitmapShader(bm,TileMode.REPEAT,TileMode.MIRROR);
paint.setShader(bs);                              //设置渲染对象
canvas.drawRoundRect(rect, 10, 10, paint);        //绘制一个使用 BitmapShader 渲染的圆角矩形图片
```

14.5.2 实现放大镜效果

【例 14.14】 本实例主要在 Android 程序中实现放大镜效果，运行程序，将显示如图 14.18 所示的运行效果，放大镜的位置跟随触摸点的改变而改变。

实例位置：光盘\MR\Instance\14\14.14

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的线性布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，用于显示自定义的绘图类。

（2）打开默认创建的 MainActivity，在该文件中，首先创建一个名称为 MyView 的内部类，该类继承自 android.view.View 类，并添加构造方法和重写 onDraw(Canvas canvas)方法，然后在 onCreate()方法中获取布局文件中添加的帧布局管理器，并将 MyView 视图添加到该帧布局管理器中。



Note



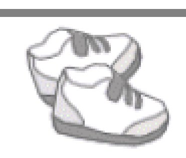
图 14.18　实现放大镜效果

（3）在内部类 MyView 中，定义源图像、放大镜图像、放大镜的半径、放大倍数、放大镜的左边距和顶边距等。其具体代码如下：

```
private Bitmap bitmap;                                          //源图像，也就是背景图像
private ShapeDrawable drawable;
private final int RADIUS = 57;                                  //放大镜的半径
private final int FACTOR = 2;                                   //放大倍数
private Matrix matrix = new Matrix();
private Bitmap bitmap_magnifier;                                //放大镜位图
private int m_left = 0;                                         //放大镜的左边距
private int m_top = 0;                                          //放大镜的顶边距
```

（4）在内部类 MyView 的构造方法中，首先获取要显示的源图像，然后创建一个 BitmapShader 对象，用于指定渲染图像，接下来再创建一个圆形的 drawable，并设置相关属性，最后获取放大镜图像，并计算放大镜的默认左边距和顶边距。其具体代码如下：

```
Bitmap bitmap_source = BitmapFactory.decodeResource(getResources(),R.drawable.source);
                                                                //获取要显示的源图像
bitmap = bitmap_source;
BitmapShader shader = new BitmapShader(Bitmap.createScaledBitmap(
        bitmap_source, bitmap_source.getWidth() * FACTOR,
        bitmap_source.getHeight() * FACTOR, true), TileMode.CLAMP,
        TileMode.CLAMP);                                        //创建 BitmapShader 对象
//圆形的 drawable
drawable = new ShapeDrawable(new OvalShape());
drawable.getPaint().setShader(shader);
drawable.setBounds(0, 0, RADIUS * 2, RADIUS * 2);               //设置圆的外切矩形
bitmap_magnifier = BitmapFactory.decodeResource(getResources(),R.drawable.magnifier);
                                                                //获取放大镜图像
m_left = RADIUS - bitmap_magnifier.getWidth() / 2;              //计算放大镜的默认左边距
m_top = RADIUS - bitmap_magnifier.getHeight() / 2;              //计算放大镜的默认顶边距
```



Note

（5）在 MyView 的 onDraw()方法中，分别绘制背景图像、放大镜图像和放大后的图像。其具体代码如下：

```
canvas.drawBitmap(bitmap, 0, 0, null);                                   //绘制背景图像
canvas.drawBitmap(bitmap_magnifier, m_left, m_top, null);               //绘制放大镜
drawable.draw(canvas);                                                    //绘制放大后的图像
```

（6）在内部类 MyView 中，重写 onTouchEvent()方法，实现当用户触摸屏幕时，放大触摸点附近的图像。其具体代码如下：

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    final int x = (int) event.getX();                                //获取当前触摸点的 X 轴坐标
    final int y = (int) event.getY();                                //获取当前触摸点的 Y 轴坐标
    //平移到绘制 shader 的起始位置
    matrix.setTranslate(RADIUS - x * FACTOR, RADIUS - y * FACTOR);
    drawable.getPaint().getShader().setLocalMatrix(matrix);
    drawable.setBounds(x - RADIUS, y - RADIUS, x + RADIUS, y + RADIUS);//设置圆的外切矩形
    m_left = x - bitmap_magnifier.getWidth() / 2;                   //计算放大镜的左边距
    m_top = y - bitmap_magnifier.getHeight() / 2;                   //计算放大镜的顶边距
    invalidate();                                                    //重绘画布
    return true;
}
```

14.5.3　忐忑的精灵

【例 14.15】 在 Eclipse 中创建 Android 项目，使用逐帧动画实现一个忐忑的精灵动画。运行本实例并单击屏幕，将播放自定义的逐帧动画，如图 14.19 所示。当动画播放时，单击屏幕，将停止动画的播放，再次单击屏幕，将继续播放动画。

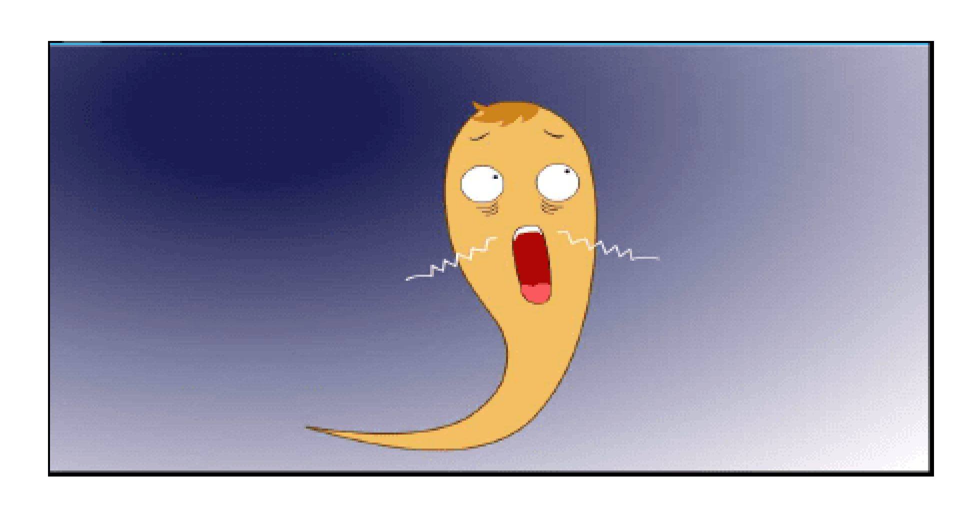


图 14.19　忐忑的精灵

实例位置：光盘\MR\Instance\14\14.15

程序的开发步骤如下：

（1）在新建项目的 res 目录中，首先创建一个名称为 anim 的目录，并在该目录中，添加一个名称为 fairy.xml 的 XML 资源文件，然后在该文件中定义一组成动画的图片资源。其具体代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/img001" android:duration="60"/>
    <item android:drawable="@drawable/img002" android:duration="60"/>
    <item android:drawable="@drawable/img003" android:duration="60"/>
    <item android:drawable="@drawable/img004" android:duration="60"/>
```



```
        <item android:drawable="@drawable/img005" android:duration="60"/>
        <item android:drawable="@drawable/img006" android:duration="60"/>
</animation-list>
```

Note

（2）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后为默认添加的线性布局管理器设置 android:id 和 android:background 属性。将 android:background 属性设置为步骤（1）中创建的动画资源，修改后的代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@anim/umbrella"
    android:id="@+id/ll"
    android:orientation="vertical" >
</LinearLayout>
```

（3）打开默认创建的 MainActivity，首先获取布局文件中添加的线性布局管理器和 AnimationDrawable 对象，然后为线性布局管理器添加事件监听器。该事件监听器中，分别调用 AnimationDrawable 对象的 start()和 stop()方法实现开始播放动画和停止播放动画的功能。其代码如下：

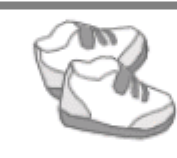
```
LinearLayout ll=(LinearLayout)findViewById(R.id.ll);           //获取布局文件中添加的线性布局管理器
//获取 AnimationDrawable 对象
final AnimationDrawable anim=(AnimationDrawable)ll.getBackground();
//为线性布局管理器添加单击事件监听器
ll.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        if(flag){
            anim.start();                                      //开始播放动画
            flag=false;
        }else{
            anim.stop();                                       //停止播放动画
            flag=true;
        }
    }
});
```

14.6　本章常见错误

在 Android 程序中使用 Canvas 绘图时，如果直接在类的 OnDraw()方法中绘制，可以正常显示绘制的图像；但是，如果把使用 Canvas 绘图的代码封装成一个函数，然后再使用类进行调用，就不能正常显示图像了，如何解决该问题呢？

使用 Canvas 绘制的图像需要显示在 View 上，而 View 只有在 onMeasure 和 onLayout 中调用，





或者手动调用 invalidate()方法时才会开始重绘，如果是将绘图代码封装成一个函数并调用，此时并没有发生重绘操作，即 View 的 draw()方法没有发生回调，所以根本无法显示绘制的图像。要解决上面描述的问题，可以使用动态绘图的方法，下面提供两种解决方法。

☑ 用数组保存每次绘制的，这个方法适用于比较简单的绘图，如绘制线段，只要在数组中保存线段的起始点即可，然后每次在 OnDraw 中全部重绘一次。

☑ 用一个 Bitmap 对象保存绘制的图像，然后将 Bitmap 绘制到 Canvas 上。其代码如下：

```
//将图像绘制到 Bitmap 上
Canvas temp=new Canvas(bitmap);
temp.drawLine(x1,y1,x2,y2,paint);
//把 Bitmap 对象绘制到 Canvas 上
canvas.drawBitmap(bitmap,0,0,paint);
```

14.7 本章小结

本章主要介绍了在 Android 中进行图形图像处理的相关技术，包括如何绘制 2D 图像、为图形添加特效，以及实现动画等内容。在介绍绘制 2D 图像时，主要介绍了如何绘制几何图形、文本、路径和图片等，在进行游戏开发时，经常需要应用到这些内容，需要读者重点掌握；在介绍实现动画效果时，主要介绍了如何实现逐帧动画和补间动画，其中，逐帧动画主要通过图片的变化来形成动画效果，而补间动画则主要体现在位置、大小、旋转度和透明度变化方面，并且只需要指定起始帧和结束帧，其他过渡帧将由系统自动计算得出。

14.8 跟我上机

参考答案：光盘\MR\跟我上机

开发一个 Android 程序，要求借助 Android 中的图形图像处理技术制作一个简单的小游戏，主要实现迷途的野猪来回奔跑的功能。具体过程为：触摸屏幕后，屏幕中的野猪将从左侧奔跑到右侧，撞到右侧的栅栏上后，再转身向左侧奔跑，直到撞上左侧的栅栏，再转身向右侧奔跑，依此类推。

实现该程序时，首先需要定义野猪做向右奔跑动作和做向左奔跑动作的逐帧动画资源文件、野猪向右侧奔跑和向左侧奔跑的补间动画资源文件。

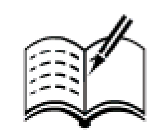
野猪做向右奔跑动作的逐帧动画资源文件名称为 motionright.xml，其代码如下：

```
<animation-list xmlns:android="http://sc hemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/pig1" android:duration="40" />
    <item android:drawable="@drawable/pig2" android:duration="40" />
</animation-list>
```



野猪做向左奔跑动作的逐帧动画资源文件名称为 motionleft.xml，其代码如下：

```
<animation-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:drawable="@drawable/pig3" android:duration="40" />
    <item android:drawable="@drawable/pig4" android:duration="40" />
</animation-list>
```



野猪向右侧奔跑的补间动画资源文件名称为 motionright.xml，其代码如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:fromXDelta="0"
        android:toXDelta="180"
        android:fromYDelta="0"
        android:toYDelta="0"
        android:duration="3000">
    </translate>
</set>
```

野猪向左侧奔跑的补间动画资源文件名称为 motionright.xml，其代码如下：

```
<set xmlns:android="http://schemas.android.com/apk/res/android" >
    <translate
        android:fromXDelta="180"
        android:toXDelta="0"
        android:fromYDelta="0"
        android:toYDelta="0"
        android:duration="3000">
    </translate>
</set>
```

然后，在 Activity 的 onCreate()方法中，获取向右奔跑的和向左奔跑的补间动画资源、ImageView 应用的逐帧动画，以及线性布局管理器，并为线性布局管理器添加触摸监听器，在重写的 onTouch()方法中，开始播放逐帧动画和“向右奔跑”的补间动画，最后为“向右奔跑”和“向左奔跑”动画添加动画监听器，在重写的 onAnimationEnd()方法中改变要使用的逐帧动画和补间动画，并播放，从而实现野猪来回奔跑的动画效果。实现野猪来回奔跑的代码参考如下：

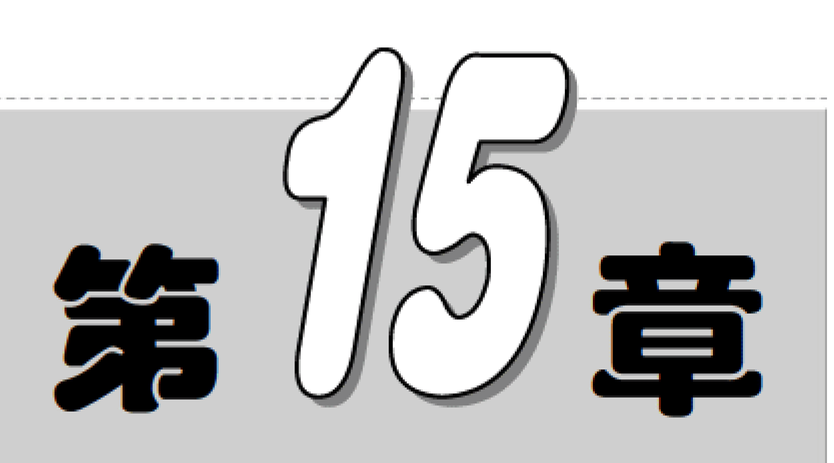
```
final ImageView iv=(ImageView)findViewById(R.id.imageView1);      //获取要应用动画效果的 ImageView
//获取“向右奔跑”动画资源
final Animation translateright=AnimationUtils.loadAnimation(this, R.anim.translateright);
//获取“向左奔跑”动画资源
final Animation translateleft=AnimationUtils.loadAnimation(this, R.anim.translateleft);
anim=(AnimationDrawable)iv.getBackground();                       //获取应用的帧动画
LinearLayout ll=(LinearLayout)findViewById(R.id.linearLayout1);   //获取线性布局管理器
Toast.makeText(this,"触摸屏幕开始播放...", Toast.LENGTH_SHORT).show();   //显示一个消息提示框
ll.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        anim.start();                                              //开始播放帧动画
```





Note

```
            iv.startAnimation(translateright);                  //播放“向右奔跑”动画
            return false;
        }
});
translateright.setAnimationListener(new AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) {}
        @Override
        public void onAnimationRepeat(Animation animation) {}
        @Override
        public void onAnimationEnd(Animation animation) {
            iv.setBackgroundResource(R.anim.motionleft);        //重新设置 ImageView 应用的帧动画
            iv.startAnimation(translateleft);                   //播放“向左奔跑”动画
            anim=(AnimationDrawable)iv.getBackground();         //获取应用的帧动画
            anim.start();                                       //开始播放帧动画
        }
});
translateleft.setAnimationListener(new AnimationListener() {
        @Override
        public void onAnimationStart(Animation animation) {}
        @Override
        public void onAnimationRepeat(Animation animation) {}
        @Override
        public void onAnimationEnd(Animation animation) {
            iv.setBackgroundResource(R.anim.motionright);       //重新设置 ImageView 应用的帧动画
            iv.startAnimation(translateright);                  //播放“向右奔跑”动画
            anim=(AnimationDrawable)iv.getBackground();         //获取应用的帧动画
            anim.start();                                       //开始播放帧动画
        }
});
```



利用 OpenGL 实现 3D 图形

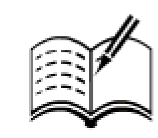
（视频讲解：56 分钟）

在现在这个网络游戏逐渐盛行的时代，2D 游戏已经不能完全满足用户的需求，3D 技术已经被广泛的应用在 PC 游戏中，3D 技术下一步肯定会向手机平台发展，而 Android 系统作为当前最流行的手机操作系统，完全内置 3D 技术——OpenGl 支持。本章将对 Android 中的 3D 技术——OpenGL 进行详细讲解。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 构建 3D 开发的基本框架
- ☐ 绘制一个 6 个面采用不同颜色的立方体
- ☐ 为立方体进行纹理贴图
- ☐ 实现一个不断旋转的立方体
- ☐ 为立方体添加光照效果
- ☐ 绘制透明且旋转的立方体
- ☐ 绘制一个不断旋转的金字塔
- ☐ 使用 Android 机器人对立方体进行纹理贴图
- ☐ 绘制一个三棱锥

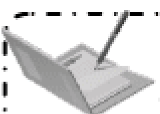




15.1 OpenGL 概述

OpenGL（Open Graphics Library）是由 SGI 公司于 1992 年发布的，一个功能强大、调用方便的底层图形库，它为编程人员提供了统一的操作，以便充分利用任何制造商提供的硬件。OpenGL 的核心实现了视区和光照等概念，并试图向开发人员隐藏大部分硬件层。

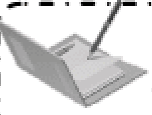
由于 OpenGL 是专门为工作站设计的，它太大了，无法安装在移动设备上。所以 Khronos Group 为 OpenGL 提供了一个子集 OpenGL ES（OpenGL for Embedded System）。OpenGL ES 是免费的、跨平台的、功能完善的 2D/3D 图形库接口 API，它专门针对多种嵌入式系统（包括手机、PDA 和游戏主机等）而设计，提供一种标准方法来描述在图形处理器或主 CPU 上渲染这些图像的底层硬件。



说明：

Khronos Group 是一个图形软硬件行业协会，该协会主要关注图形和多媒体方向的开放标准。

OpenGL ES 去除了 OpenGL 中的 glBegin/glEnd、四边形（GL_QUADS）、多边形（GL_POLYGONS）等复杂图元等许多非绝对必要的特性。经过多年发展，目前的 OpenGL ES 现在主要有 OpenGL ES 1.x（针对固定管线硬件）和 OpenGL ES 2.x（针对可编程管线硬件）两个版本。OpenGL ES 1.0 是以 OpenGL 1.3 规范为基础的，OpenGL ES 1.1 是以 OpenGL 1.5 规范为基础的，OpenGL ES 2.0 则是参照 OpenGL 2.0 规范定义的，它补充和修改了 OpenGL ES 1.1 标准着色器语言及 API，将 OpenGL ES 1.1 中所有可以用着色器程序替换的功能全部删除了，这样可以节约移动设备的开销及电力消耗。



说明：

OpenGL ES 可以应用于很多主流移动平台上，包括 Android、Symbian 和 iPhone 等。

Android 为 OpenGL 提供了相应的支持，它专门为支持 OpenGL 提供了 android.opengl 包。在该包中，GLES10 类是为支持 OpenGL ES 1.0 而提供的，GLES11 类是为支持 OpenGL ES 1.1 而提供的，GLES20 类是为支持 OpenGL ES 2.0 而提供的。其中，OpenGL ES 2.0 是从 Android 2.2（API Level 8）版本才开始使用的。

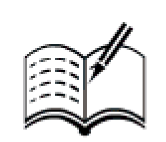


说明：

如果用户的应用只支持 OpenGL ES 2.0，必须在该项目的 AndroidManifest.xml 文件中添加下列设置。

```
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```





15.2　绘制 3D 图形

OpenGL ES 一个最常用的功能就是绘制 3D 图形。要绘制 3D 图形，大致可以分为两个步骤，下面分别进行讲解。

15.2.1　构建 3D 开发的基本框架

构建一个 3D 开发的基本框架大致可以分为以下几个步骤。

（1）创建一个 Activity，并指定该 Activity 显示的内容是一个指定了 Renderer 对象的 GLSurfaceView 对象。例如，创建一个名称为 MainActivity 的 Activity，在重写的 onCreate()方法中创建一个 GLSurfaceView 对象，并为其指定使用的 Renderer 对象，再将其设置为 Activity 要显示的内容，可以使用下面的代码。

```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	GLSurfaceView mGLView = new GLSurfaceView(this);//创建一个 GLSurfaceView 对象
	mGLView.setRenderer(new CubeRenderer());  //为 GLSurfaceView 指定使用的 Renderer 对象
	setContentView(mGLView);                  //设置 Activity 显示的内容为 GLSurfaceView 对象
}
```

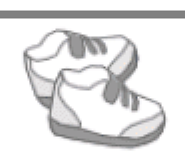
通常情况下，考虑到当 Activity 恢复和暂停时，GLSurfaceView 对象也恢复或者暂停，还要重写 Activity 的 onResume()和 onPause()方法。例如，如果一个 Activity 使用的 GLSurfaceView 对象为 mGLView，那么，可以使用以下代码重写 onResume()和 onPause()方法。

```
@Override
protected void onResume() {
	super.onResume();
	mGLView.onResume();
}
@Override
protected void onPause() {
	super.onPause();
	mGLView.onPause();
}
```

（2）创建实现 GLSurfaceView.Renderer 接口的类。在创建该类时，需要实现接口中的以下 3 个方法。

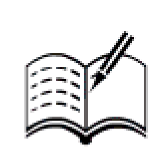
☑ public void onSurfaceCreated(GL10 gl, EGLConfig config)：当 GLSurfaceView 被创建时回调该方法。

☑ public void onDrawFrame(GL10 gl)：Renderer 对象调用该方法绘制 GLSurfaceView 的当



前帧。

☑ public void onSurfaceChanged(GL10 gl, int width, int height)：当 GLSurfaceView 的大小改变时回调该方法。



Note

例如，创建一个实现 GLSurfaceView.Renderer 接口的类 EmptyRenderer，并实现 onSurfaceCreated()、onDrawFrame()和 onSurfaceChanged()方法，为窗体设置背景颜色。其具体代码如下：

```
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;
public class EmptyRenderer implements GLSurfaceView.Renderer {
	public void onSurfaceCreated(GL10 gl, EGLConfig config) {
		//设置窗体的背景颜色
		gl.glClearColor(0.7f, 0.7f, 0.9f, 1.0f);
	}
	public void onDrawFrame(GL10 gl) {
		//重设背景颜色
		gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
	}
	public void onSurfaceChanged(GL10 gl, int width, int height) {
		gl.glViewport(0, 0, width, height);
	}
}
```

当窗口被创建时，需要调用 onSurfaceCreated()方法进行一些初始化操作。onSurfaceCreated()方法有一个 GL10 类型的参数 gl，gl 相当于 OpenGL ES 的画笔。通过它提供的方法不仅可以绘制 3D 图形，也可以对 OpenGL 进行初始化。表 15.1 给出了 GL10 提供的用于进行初始化的方法，对于 GL10 提供的用于绘制 3D 图形的方法将在 15.2.2 节进行介绍。

表 15.1　GL10 提供的用于进行初始化的方法

方　法	描　述
glClearColor(float red, float green, float blue, float alpha)	用于指定清除屏幕时使用的颜色，4 个参数分别用于设置红、绿、蓝和透明度的值，值的范围是 0.0f～1.0f
glDisable(int cap)	用于禁用 OpenGL ES 某个方面的特性。例如，要关闭抗抖动功能，可以使用“gl.glDisable(GL10.GL_DITHER);”语句
glEnable(int cap)	用于启用 OpenGL ES 某个方面的特性
glFrustumf(float left, float right, float bottom, float top, float zNear, float zFar)	用于设置透视视窗的空间大小
glHint(int target, int mode)	用于对 OpenGL ES 某个方面进行修正
glLoadIdentity()	用于初始化单位矩阵
glMatrixMode(int mode)	用于设置视图的矩阵模式。通常可以使用 GL10.GL_MODELVIEW 和 GL10.GL_PROJECTION 两个常量值
glShadeModel(int mode)	用于设置 OpenGL ES 的阴影模式。例如，要设置为平滑模式，可以使用“gl.glShadeModel(GL10.GL_SMOOTH);”语句
glViewport(int x, int y, int width, int height)	用于设置 3D 场景的大小



15.2.2　绘制一个模型

在基本框架构建完成后，就可以在该框架的基础上绘制 3D 模型了。在 OpenGL ES 中，任何模型都会被分解为三角形。下面将以绘制一个 2D 的三角形为例介绍绘制 3D 模型的基本步骤。

（1）在 onSurfaceCreated()方法中，定义顶点坐标数组。例如，要绘制一个二维的三角形，可以使用以下代码定义顶点坐标数组。

```
private final IntBuffer mVertexBuffer;
public GLTriangle() {
        int one = 65536;
        int vertices[] = {
                0, one, 0,                //上顶点
                -one, -one, 0,            //左下点
                one, -one, 0              //右下点
        };
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());
        mVertexBuffer = vbb.asIntBuffer();
        mVertexBuffer.put(vertices);
        mVertexBuffer.position(0);
}
```

说明：

在默认的情况下，OpenGL ES 采取的坐标是（X,Y,Z），[0,0,0]表示 GLSurfaceView 的中心；[1,1,0]表示 GLSurfaceView 的右上角；[−1,−1,0]表示 GLSurfaceView 的左下角。

（2）在 onSurfaceCreated()方法中，应用以下代码启用顶点坐标数组。

```
gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);          //启用顶点坐标数组
```

（3）在 onDrawFrame()方法中，应用步骤（1）定义的顶点坐标数组绘制图形。例如，要绘制一个三角形可以使用下面的代码。

```
gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);     //为画笔指定顶点坐标数据
gl.glColor4f(1, 0, 0, 0.5f);                                //设置画笔颜色
gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 3);              //绘制图形
```

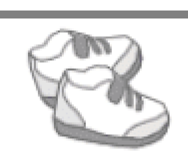
在了解了应用 OpenGL ES 绘制 3D 图形的基本步骤后，下面将介绍一个具体的实例来介绍如何绘制一个立方体。

【例 15.1】 在 Eclipse 中创建 Android 项目，绘制一个 6 个面采用不同颜色的立方体。

实例位置：光盘\MR\Instance\15\15.1

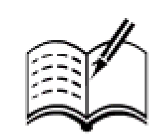
程序的开发步骤如下：

（1）在默认创建的 MainActivity 中，创建一个 GLSurfaceView 类型的成员变量。其关键代



码如下：

```
private GLSurfaceView mGLView;
```



Note

（2）在重写的 onCreate()方法中，首先创建一个 GLSurfaceView 对象，然后为 GLSurfaceView 指定使用的 Renderer 对象，最后再设置 Activity 显示的内容为 GLSurfaceView 对象。其关键代码如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      mGLView = new GLSurfaceView(this);          //创建一个 GLSurfaceView 对象
      mGLView.setRenderer(new CubeRenderer()); //为 GLSurfaceView 指定使用的 Renderer 对象
      setContentView(mGLView);                    //设置 Activity 显示的内容为 GLSurfaceView 对象
}
```

（3）重写 onResume()和 onPause()方法。其具体代码如下：

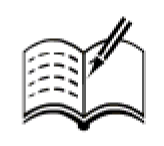
```
@Override
protected void onResume() {
      super.onResume();
      mGLView.onResume();
}
@Override
protected void onPause() {
      super.onPause();
      mGLView.onPause();
}
```

（4）创建一个实现 GLSurfaceView.Renderer 接口的类 CubeRenderer，并实现 onSurfaceCreated()、onDrawFrame()和 onSurfaceChanged()方法。其具体代码如下：

```
import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;
import android.opengl.GLSurfaceView;
public class CubeRenderer implements GLSurfaceView.Renderer {
      @Override
      public void onDrawFrame(GL10 gl) {
      }
      @Override
      public void onSurfaceChanged(GL10 gl, int width, int height) {
      }
      @Override
      public void onSurfaceCreated(GL10 gl, EGLConfig config) {
      }
}
```

（5）在 onSurfaceCreated()方法中，应用以下代码进行初始化操作，主要包括设置窗体背景颜色、启用顶点坐标数组、关闭抗抖动功能、设置系统对透视进行修正、设置阴影平滑模式、启





用深度测试及设置深度测试的类型等。其具体代码如下：

```
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    gl.glClearColor(0.7f, 0.9f, 0.9f, 1.0f);                //设置窗体背景颜色
    gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);           //启用顶点坐标数组
    gl.glDisable(GL10.GL_DITHER);                           //关闭抗抖动
    //设置系统对透视进行修正
    gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
    gl.glShadeModel(GL10.GL_SMOOTH);                        //设置阴影平滑模式
    gl.glEnable(GL10.GL_DEPTH_TEST);                        //启用深度测试
    gl.glDepthFunc(GL10.GL_LEQUAL);                         //设置深度测试的类型
}
```

说明：

深度测试就是让 OpenGL ES 负责跟踪每个物体在 Z 轴上的深度，这样可避免后面的物体遮挡前面的物体。

（6）在 onSurfaceChanged()方法中，首先设置 OpenGL 场景的大小，并计算透视视窗的宽度、高度比，然后将当前矩阵模式设为投影矩阵，再初始化单位矩阵，最后设置透视视窗的空间大小。其具体代码如下：

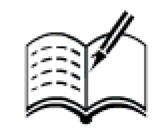
```
public void onSurfaceChanged(GL10 gl, int width, int height) {
    gl.glViewport(0, 0, width, height);                     //设置 OpenGL 场景的大小
    float ratio = (float) width / height;                   //计算透视视窗的宽度、高度比
    gl.glMatrixMode(GL10.GL_PROJECTION);                    //将当前矩阵模式设为投影矩阵
    gl.glLoadIdentity();                                    //初始化单位矩阵
    GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);          //设置透视视窗的空间大小
}
```

（7）在 onDrawFrame()方法中，首先清除颜色缓存和深度缓存，并设置使用模型矩阵进行变换，然后初始化单位矩阵，再设置视点，并旋转总坐标系，最后绘制立方体。其具体代码如下：

```
public void onDrawFrame(GL10 gl) {
     //清除颜色缓存和深度缓存
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    gl.glMatrixMode(GL10.GL_MODELVIEW);                     //设置使用模型矩阵进行变换
    gl.glLoadIdentity();                                    //初始化单位矩阵
    //当使用 GL_MODELVIEW 模式时，必须设置视点，也就是观察点
    GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
    gl.glRotatef(1000, -0.1f, -0.1f, 0.05f);                //旋转总坐标系
    cube.draw(gl);                                          //绘制立方体
}
```

（8）创建一个用于绘制立方体模型的 Java 类，名称为 GLCube，在该类中，首先定义一个用于记录顶点坐标数据缓冲的成员变量。其关键代码如下：

```
public class GLCube {
    private final IntBuffer mVertexBuffer;                    //顶点坐标数据缓冲
}
```

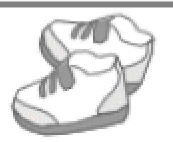

Note

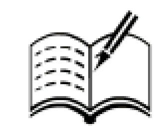
（9）定义 GLCube 类的构造方法，在构造方法中创建一个记录顶点位置的数组，并根据该数组创建顶点坐标数据缓冲，其具体代码如下：

```
public GLCube() {
    int one = 65536;
    int half = one / 2;
    int vertices[] = {
            //前面
            -half, -half, half, half, -half, half,
            -half, half, half, half, half, half,
            //背面
            -half, -half, -half, -half, half, -half,
            half, -half, -half, half, half, -half,
            //左面
            -half, -half, half, -half, half, half,
            -half, -half, -half, -half, half, -half,
            //右面
            half, -half, -half, half, half, -half,
            half, -half, half, half, half, half,
            //上面
            -half, half, half, half, half, half,
            -half, half, -half, half, half, -half,
            //下面
            -half, -half, half, -half, -half, -half,
            half, -half, half, half, -half, -half,
        };                                                    //定义顶点位置
    //创建顶点坐标数据缓冲
    ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
    vbb.order(ByteOrder.nativeOrder());                       //设置字节顺序
    mVertexBuffer = vbb.asIntBuffer();                        //转换为 int 型缓冲
    mVertexBuffer.put(vertices);                              //向缓冲中放入顶点坐标数据
    mVertexBuffer.position(0);                                //设置缓冲区的起始位置
}
```

（10）在 GLCube 类中，编写用于绘制立方体的 draw()方法。在该方法中，首先为画笔指定顶点坐标数组，然后分别绘制立方体的 6 个面，每个面使用的颜色是不同的。draw()方法的具体代码如下：

```
    public void draw(GL10 gl) {
    gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);   //为画笔指定顶点坐标数据
    //绘制 FRONT 和 BACK 两个面
    gl.glColor4f(1, 0, 0, 1);
    gl.glNormal3f(0, 0, 1);
    gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);            //绘制图形
    gl.glColor4f(1, 0, 0.5f, 1);
```






```
        gl.glNormal3f(0, 0, -1);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 4);              //绘制图形
        //绘制 LEFT 和 RIGHT 两个面
        gl.glColor4f(0, 1, 0, 1);
        gl.glNormal3f(-1, 0, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 8, 4);              //绘制图形
        gl.glColor4f(0, 1, 0.5f, 1);
        gl.glNormal3f(1, 0, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 12, 4);             //绘制图形
        //绘制 TOP 和 BOTTOM 两个面
        gl.glColor4f(0, 0, 1, 1);
        gl.glNormal3f(0, 1, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 16, 4);             //绘制图形
        gl.glColor4f(0, 0, 0.5f, 1);
        gl.glNormal3f(0, -1, 0);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 20, 4);             //绘制图形
}
```

（11）打开 CubeRenderer 类，在该类中创建一个代表立方体对象的成员变量，并为 CubeRenderer 类创建无参的构造方法，在该构造方法中，实例化立方体对象。其关键代码如下：

```
private final GLCube cube;                                          //立方体对象
public CubeRenderer() {
        cube = new GLCube();                                        //实例化立方体对象
}
```

实例的运行效果如图 15.1 所示。

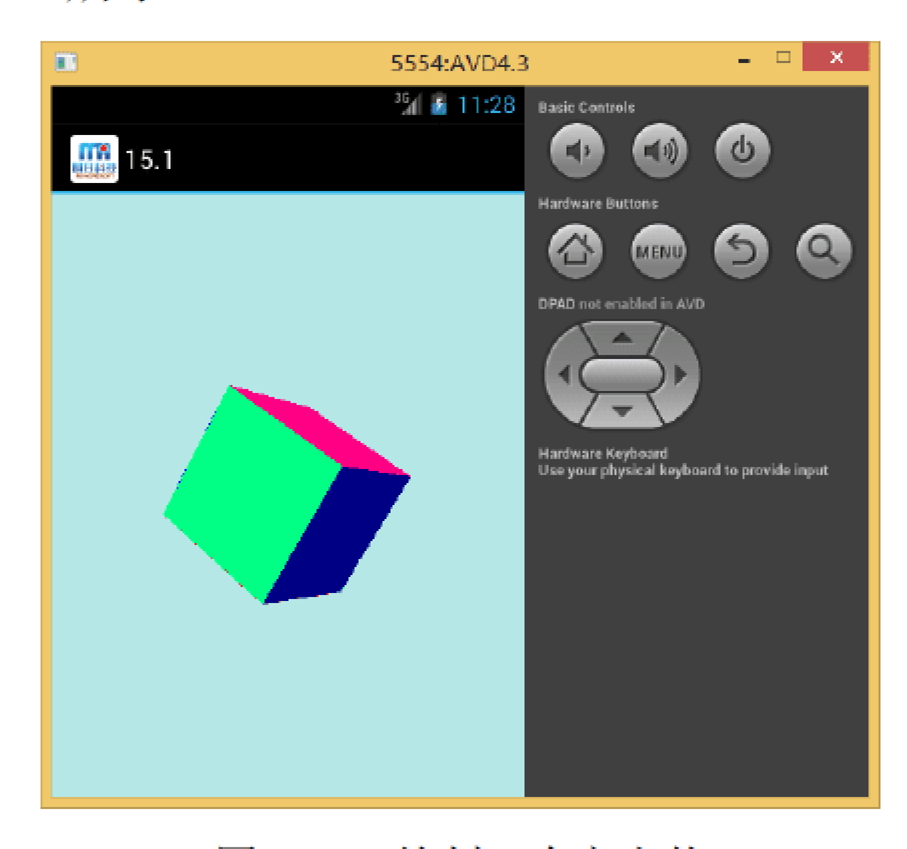


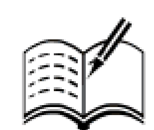
图 15.1　绘制一个立方体

15.3　添 加 效 果

15.2 节中介绍了如何绘制 3D 模型，在实际应用开发时，经常需要为其添加纹理贴图、光照和旋转等效果。本节将介绍如何为 3D 模型添加纹理贴图、旋转、光照以及透明效果等。



15.3.1 应用纹理贴图



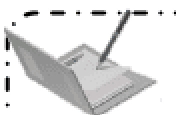
Note

为了让 3D 图形更加逼真，需要为这些 3D 图形应用纹理贴图。例如，要在场景中放置一个木箱，那么就需要为场景中绘制的立方体应用木材纹理进行贴图。为 3D 模型添加纹理贴图大致可以分为以下 3 个步骤：

（1）设置贴图坐标的数组信息，这与设置顶点坐标数组类似。

（2）设置启用贴图坐标数组。

（3）调用 GL10 的 texImage2D()方法生成纹理。



说明：

在使用纹理贴图时，需要准备一张纹理图片，建议该图片的长宽是 2 的 N 次方，例如，可以是 256×256 的图片，也可以是 512×512 的图片。

【例 15.2】 在例 15.1 的基础上为绘制的立方体进行纹理贴图。

实例位置：光盘\MR\Instance\15\15.2

程序的开发步骤如下：

（1）打开 GLCube 类文件，在该类中定义用于保存纹理贴图数据缓冲的成员变量。其具体代码如下：

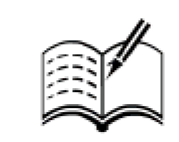
```
private IntBuffer mTextureBuffer;                          //纹理贴图数据缓冲
```

（2）打开 GLCube 类文件，在构造方法中，定义贴图坐标数组，并根据该数组创建贴图坐标数据缓冲。其具体代码如下：

```
int texCoords[] = {
        //前面
        0, one, one, one, 0, 0, one, 0,
        //后面
        one, one, one, 0, 0, one, 0, 0,
        //左面
        one, one, one, 0, 0, one, 0, 0,
        //右面
        one, one, one, 0, 0, one, 0, 0,
        //上面
        one, 0, 0, 0, one, one, 0, one,
        //下面
        0, 0, 0, one, one, 0, one, one, };                 //定义贴图坐标数组
ByteBuffer tbb = ByteBuffer.allocateDirect(texCoords.length * 4);
tbb.order(ByteOrder.nativeOrder());                        //设置字节顺序
mTextureBuffer = tbb.asIntBuffer();                        //转换为 int 型缓冲
mTextureBuffer.put(texCoords);                             //向缓冲中放入贴图坐标数组
mTextureBuffer.position(0);                                //设置缓冲区的起始位置
```

（3）在 GLCube 类的 draw()方法的最后，应用 GL10 的 glTexCorrdPointer()方法为画笔指定





贴图坐标数据。其关键代码如下：

```
gl.glTexCoordPointer(2, GL10.GL_FIXED, 0, mTextureBuffer);//为画笔指定贴图坐标数据
```

（4）编写 loadTexture()方法，用于进行纹理贴图。其具体代码如下：

```
/**
 *
 * 功能：进行纹理贴图
 *
 * @param gl
 * @param context
 * @param resource
 */
void loadTexture(GL10 gl, Context context, int resource) {
      Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
                  resource);                                          //加载位图
      GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);            //使用图片生成纹理
      bmp.recycle();                                                 //释放资源
}
```

（5）打开 CubeRenderer 类文件，在 onSurfaceCreated()方法中添加以下代码，首先启用贴图坐标数组，然后启用纹理贴图，最后调用 GLCube 类的 loadTexture()方法进行纹理贴图。

```
gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);            //启用贴图坐标数组
gl.glEnable(GL10.GL_TEXTURE_2D);                                  //启用纹理贴图
cube.loadTexture(gl, context, R.drawable.mr);                     //进行纹理贴图
```

运行本实例，将显示如图 15.2 所示的运行效果。

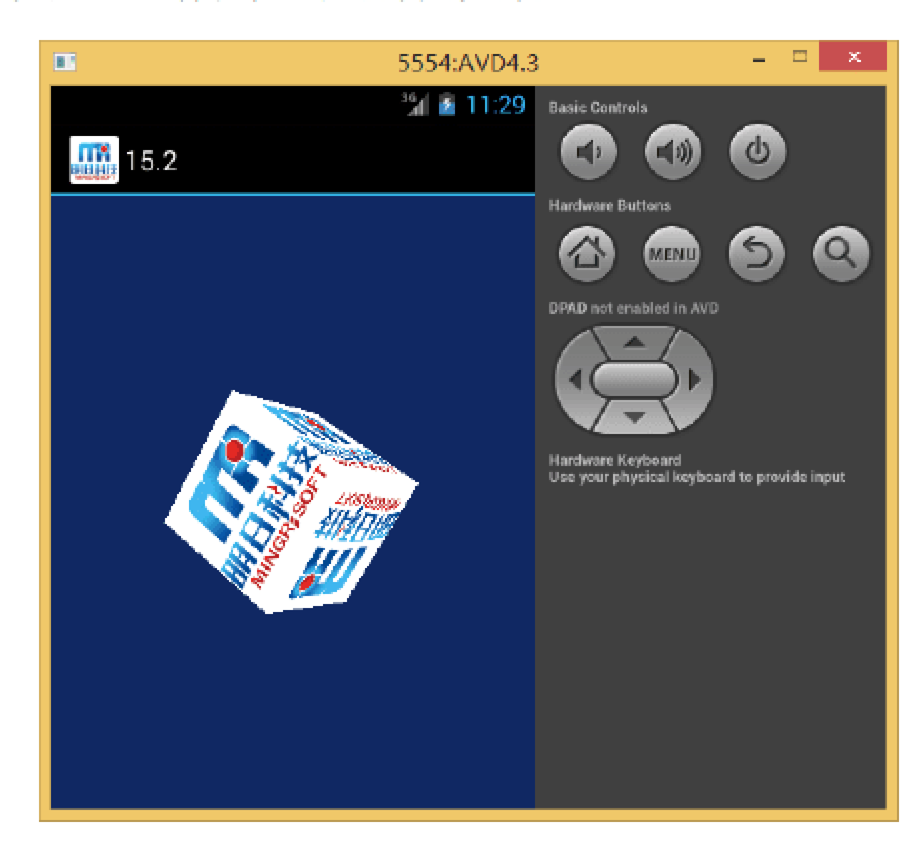
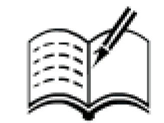


图 15.2　为立方体进行纹理贴图

15.3.2　旋转

到目前为止，绘制的 3D 物体还是静止的，为了更好地看到 3D 效果，还可以为其添加旋转





Note

效果，这样就可达到动画效果了。要实现旋转比较简单，只需要使用 GL10 的 glRotatef()方法不断地旋转要放置的对象即可。glRotatef()方法的语法格式如下：

```
glRotatef(float angle, float x, float y, float z)
```

其中，参数 angle 通常是一个变量，表示对象转过的角度；参数 x 表示 X 轴的旋转方向（值为 1 表示顺时针，−1 表示逆时针方向，0 表示不旋转）；参数 y 表示 Y 轴的旋转方向（值为 1 表示顺时针，−1 表示逆时针方向，0 表示不旋转）；参数 z 表示 Z 轴的旋转方向（值为 1 表示顺时针，−1 表示逆时针方向，0 表示不旋转）。

例如，要将对象经过 X 轴旋转 n 角度，可以使用下面的代码。

```
gl.glRotatef(n, 1, 0, 0);
```

【例 15.3】 在例 15.2 的基础上实现一个不断旋转的立方体。

实例位置：光盘\MR\Instance\15\15.3

程序的开发步骤如下：

（1）打开 CubeRenderer 类文件，在该类中定义，用于保存开始时间的成员变量。其具体代码如下：

```
private long startTime;                                    //保存开始时间
```

（2）在构造方法中，为成员变量 startTime 赋初始值为当前时间。其具体代码如下：

```
startTime=System.currentTimeMillis();
```

（3）在 onDrawFrame()方法绘制立方体的代码之前，添加以下代码，完成旋转立方体的操作。

```
//旋转
long elapsed = System.currentTimeMillis() - startTime;          //计算逝去的时间
gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);                 //在 Y 轴上旋转 30 度
gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);                 //在 X 轴上旋转 15 度
```

运行本实例，将显示如图 15.3 所示的运行效果。

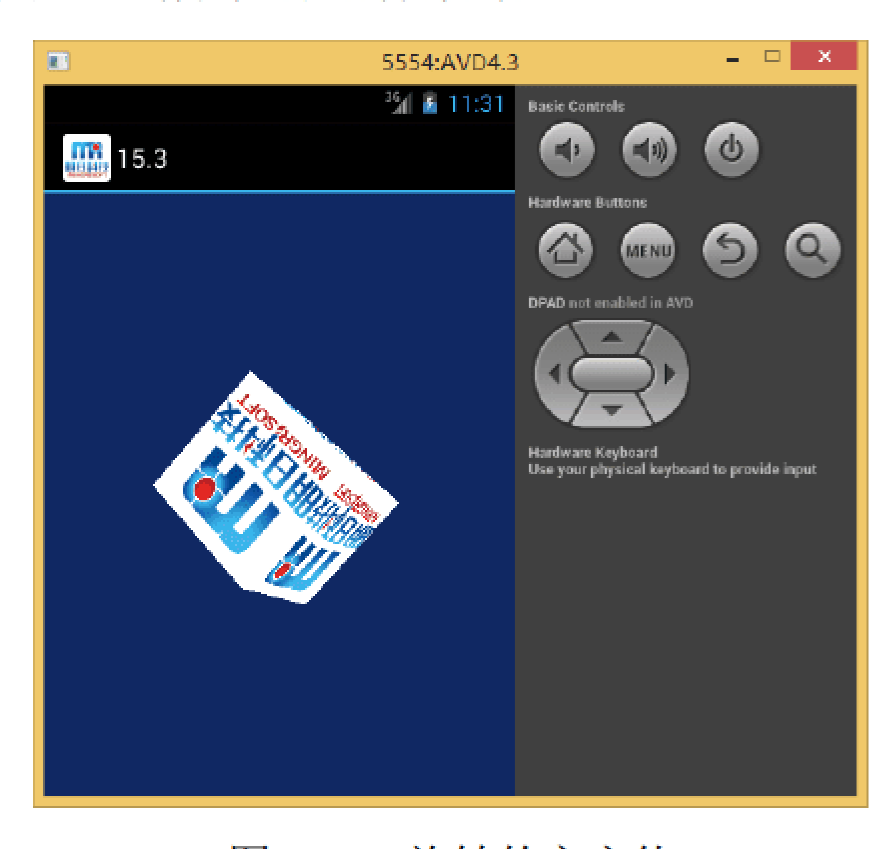
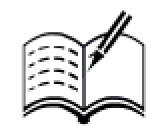


图 15.3　旋转的立方体





15.3.3 光照效果

为了使程序效果更加美观、逼真，还可以让其模拟光照效果。在为物体添加光照效果前，先来了解一下 3D 图形支持的光照类型。所有的 3D 图形都支持以下 3 种光照类型。

☑ 环境光：一种普通的光线，光线会照亮整个场景，即使对象背对着光线也可以。

☑ 散射光：柔和的方向性光线。例如，荧光板上发出的光线就是这种散射光。场景中的大部分光线通常来源于散射光源。

☑ 镜面高光：耀眼的光线，通常来源于明亮的点光源。与有光泽的材料结合使用时，这种光会带来高光效果，增加场景的真实感。

在 OpenGL 中，添加光照效果，通常分为以下两个步骤进行。

1. 光线

在定义光照效果时，通常需要定义光线，也就是为场景添加光源，这可以通过 GL10 提供的 glLightfv()方法实现。glLightfv()方法的语法格式如下：

```
glLightfv(int light, int pname, float[] params, int offset)
```

其中，参数 light 表示光源的 ID，当程序中包含多个光源时，可以通过这个 ID 来区分光源；参数 pname 表示光源的类型（参数值为 GL10.GL_AMBIENT 表示环境光，参数值为 GL10.GL_DIFFUSE 表示散射光）；参数 params，表示光源数组；参数 offset，表示偏移量。

例如，要定义一个发出白色的全方向的光源，可以使用下面的代码。

```
float lightAmbient[]=new float[]{0.2f,0.2f,0.2f,1};                    //定义环境光
float lightDiffuse[]=new float[]{1,1,1,1};                             //定义散射光
float lightPos[]=new float[]{1,1,1,1};                                 //定义光源的位置
gl.glEnable(GL10.GL_LIGHTING);                                         //启用光源
gl.glEnable(GL10.GL_LIGHT0);                                           //启用 0 号光源
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient,0);         //设置环境光
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);        //设置散射光
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);           //设置光源的位置
```

注意：

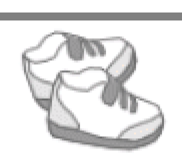
在定义和设置光源后，还需要使用 glEnable()方法启用光源，否则，设置的光源将不起作用。

2. 被照射的物体

在定义光照效果时，通常需要定义被照射物体的制作材料，因为不同材料的光线反射情况是不同的。使用 GL10 提供的 glMaterialfv()方法可以设置材质的环境光和散射光。glMaterialfv()方法的语法格式如下：

```
glMaterialfv(int face, int pname, float[] params, int offset)
```

其中，参数 face 表示是为正面还是背面材质设置光源；参数 pname 表示光源的类型（参数




Note

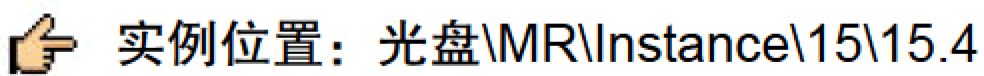
值为 GL10.GL_AMBIENT 表示环境光，参数值为 GL10.GL_DIFFUSE 表示散射光）；参数 params 表示光源数组；参数 offset 表示偏移量。

例如，定义一个不是很亮的纸质的物体，可以使用下面的代码。

```
float matAmbient[]=new float[]{1,1,1,1};                                              //定义材质的环境光
float matDiffuse[]=new float[]{1,1,1,1};                                              //定义材质的散射光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, matAmbient,0);  //设置材质的环境光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, matDiffuse,0);  //设置材质的散射光
```

下面通过一个具体的实例来说明为物体添加光照效果的具体步骤。

【例 15.4】 在例 15.3 的基础上实现为旋转的立方体添加光照效果的功能。


实例位置：光盘\MR\Instance\15\15.4

程序的开发步骤如下：

（1）打开 CubeRenderer 类文件，在 onSurfaceCreated()方法中为被照射的物体设置材质，首先定义材质的环境光和散射光，然后设置材质的环境光和散射光。其具体代码如下：

```
float matAmbient[]=new float[]{1,1,1,1};                                              //定义材质的环境光
float matDiffuse[]=new float[]{1,1,1,1};                                              //定义材质的散射光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_AMBIENT, matAmbient,0);   //设置材质的环境光
gl.glMaterialfv(GL10.GL_FRONT_AND_BACK, GL10.GL_DIFFUSE, matDiffuse,0);  //设置材质的散射光
```

（2）在 onSurfaceCreated()方法中添加场景光线，首先定义环境光和散射光，并定义光源的位置，然后启用光源和 0 号光源，最后设置环境光、散射光和光源的位置。其具体代码如下：

```
float lightAmbient[]=new float[]{0.2f,0.2f,0.2f,1};                                  //定义环境光
float lightDiffuse[]=new float[]{1,1,1,1};                                            //定义散射光
float lightPos[]=new float[]{1,1,1,1};                                                //定义光源的位置
gl.glEnable(GL10.GL_LIGHTING);                                                        //启用光源
gl.glEnable(GL10.GL_LIGHT0);                                                          //启用 0 号光源
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_AMBIENT, lightAmbient,0);                       //设置环境光
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_DIFFUSE, lightDiffuse, 0);                      //设置散射光
gl.glLightfv(GL10.GL_LIGHT0, GL10.GL_POSITION, lightPos, 0);                         //设置光源的位置
```

运行本实例，将显示如图 15.4 所示的运行效果。

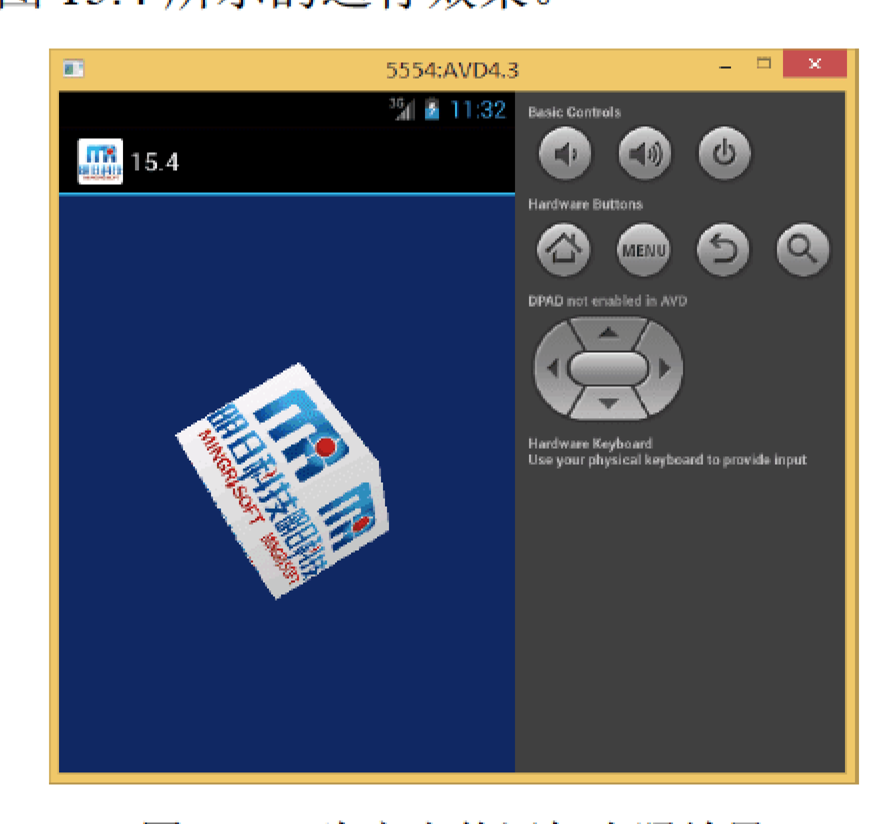
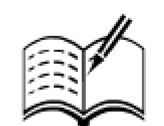


图 15.4　为立方体添加光照效果





15.3.4　透明效果

在游戏中，经常需要应用透明效果，使用 OpenGL ES 实现简单的透明效果也比较简单，只需要应用以下代码就可以实现。

```
gl.glDisable(GL10.GL_DEPTH_TEST);                    //关闭深度测试
gl.glEnable(GL10.GL_BLEND);                          //打开混合
gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE);  //使用 alpha 通道值进行混色，从而达到透明效果
```



说明：

实现透明效果时，需要关闭深度测试，并且打开混合效果，然后才能使用 GL10 类的 glBlendFunc()方法进行混色，从而达到透明效果。

下面通过一个具体的实例来说明实现透明效果的具体步骤。

【例 15.5】 在例 15.4 的基础上制作一个透明的、不断旋转的立方体。

实例位置：光盘\MR\Instance\15\15.5

程序的开发步骤如下：

打开 CubeRenderer 类文件，在 onSurfaceCreated()方法中为立方体添加透明效果，首先关闭深度测试，然后打开混合效果，最后再使用 alpha 通道值进行混色，从而达到透明效果。其具体代码如下：

```
gl.glDisable(GL10.GL_DEPTH_TEST);                    //关闭深度测试
gl.glEnable(GL10.GL_BLEND);                          //打开混合
gl.glBlendFunc(GL10.GL_SRC_ALPHA, GL10.GL_ONE); //使用 alpha 通道值进行混色，从而达到透明效果
```

运行本实例，将显示如图 15.5 所示的运行效果。

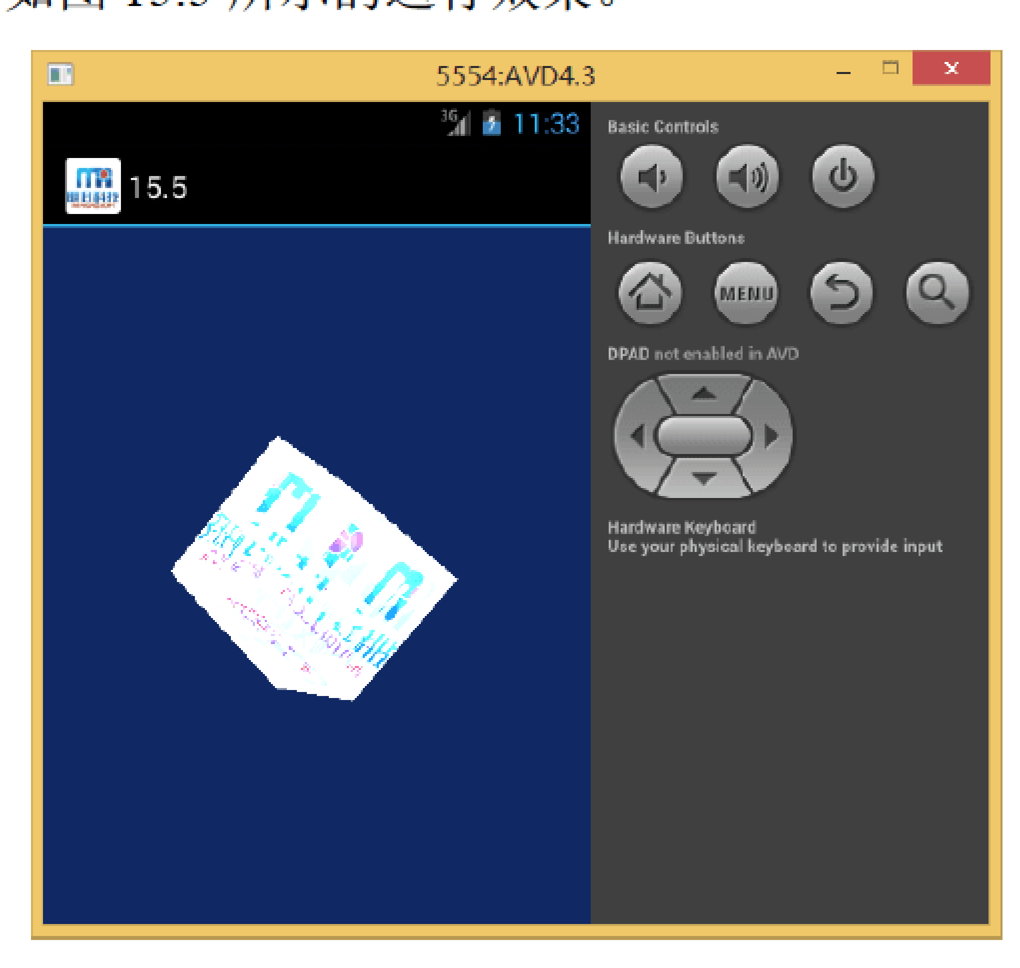
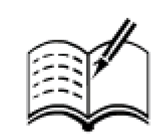


图 15.5　透明且旋转的立方体





Note

15.4 综合应用

15.4.1 绘制一个不断旋转的金字塔

【例 15.6】 使用 OpenGL ES 可以很方便地绘制一个不断旋转的金字塔，也就是一个四棱锥，本实例要求绘制一个从顶到底渐变的、不断旋转的金字塔。程序的运行效果如图 15.6 所示。

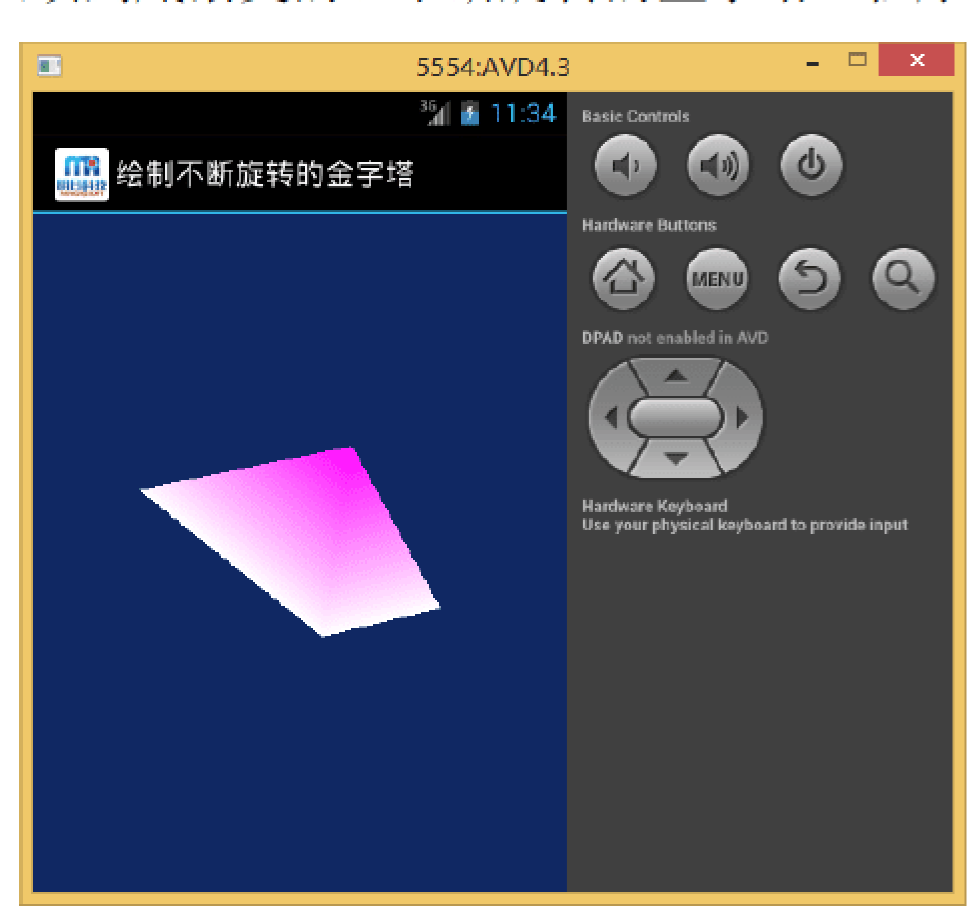


图 15.6　绘制一个不断旋转的金字塔

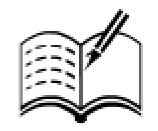
实例位置：光盘\MR\Instance\15\15.6

程序的开发步骤如下：

（1）绘制金字塔，首先需要定义金字塔的顶点坐标位置和各个切面的颜色，并使用 GL10 对象的相关方法绘制金字塔；然后通过自定义类实现 GLSurfaceView.Renderer 接口的 CubeRenderer 类，并实现其中的 onSurfaceCreated()、onDrawFrame()和 onSurfaceChanged()方法，在这 3 个方法中分别对金字塔的背景颜色、场景、旋转等进行设置。定义金字塔顶点坐标及绘制金字塔的代码如下：

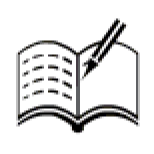
```
public class GLPyramid {
	private final IntBuffer mVertexBuffer;                    //顶点坐标数据缓冲
	private IntBuffer mColorBuffer;                           //纹理贴图数据缓冲
	public GLPyramid() {
		int one = 65535;
		int vertices[] = {
				//底面
				 -one,   0, one,
				 one,   0,   one,
				 -one, 0, -one,
				 one, 0, -one,
```






```
                one,0,one,one,0,-one,0,one,0,
                0,one,0,one,0,-one,-one,0,-one,
                -one,0,-one,-one,0,one,0,one,0,
                0,one,0,-one,0,one,one,0,one
        };                                                          //定义顶点位置
        //创建顶点坐标数据缓冲
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());                         //设置字节顺序
        mVertexBuffer = vbb.asIntBuffer();                          //转换为 int 型缓冲
        mVertexBuffer.put(vertices);                                //向缓冲中放入顶点坐标数据
        mVertexBuffer.position(0);                                  //设置缓冲区的起始位置
        /*********************** 颜色 ***********************************************/
            int colors[] = {
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, 0, one, one,
                one, 0, one, one,
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, one, one, one,
                one, 0, one, one,
                one, 0, one, one,
                one, one, one, one,
                one, one, one, one,
            };                                                      //定义颜色坐标数据
        ByteBuffer tbb = ByteBuffer.allocateDirect(colors.length * 4);
        tbb.order(ByteOrder.nativeOrder())                          //设置字节顺序
        mColorBuffer = tbb.asIntBuffer();                           //转换为 int 型缓冲
        mColorBuffer.put(colors);                                   //向缓冲中放入颜色坐标数据
        mColorBuffer.position(0);                                   //设置缓冲区的起始位置
        /*******************************************************************************/
    }
    public void draw(GL10 gl) {
        gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);     //为画笔指定顶点坐标数据
        gl.glEnableClientState(GL10.GL_COLOR_ARRAY);
        //绘制底面
        gl.glColorPointer(4, GL10.GL_FIXED, 0, mColorBuffer);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 4);              //绘制图形
        //绘制 4 个侧面
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 4, 12);             //绘制图形
    }
}
```

（2）实现 GLSurfaceView.Renderer 接口的 CubeRenderer 类，并实现其中的 onSurfaceCreated()、onDrawFrame()和 onSurfaceChanged()方法的代码如下：


Note

```
public class PyramidRenderer implements GLSurfaceView.Renderer {
    private final GLPyramid pyramid;                          //四棱锥对象
    private long startTime;                                   //定义变量保存开始时间
    public PyramidRenderer(Context context) {
        pyramid = new GLPyramid();                            //实例化四棱锥对象
        startTime=System.currentTimeMillis();
    }
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.08f, 0.16f, 0.39f, 1.0f);           //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);         //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                         //关闭抗抖动
        gl.glShadeModel(GL10.GL_SMOOTH);                      //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                      //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                       //设置深度测试的类型
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);                   //设置 OpenGL 场景的大小
        gl.glMatrixMode(GL10.GL_PROJECTION);                  //将当前矩阵模式设为投影矩阵
        float ratio = (float) width / height;                 //计算透视视窗的宽度、高度比
        gl.glLoadIdentity();                                  //初始化单位矩阵
        GLU.gluPerspective(gl, 60.0f, ratio, 1, 100f);        //设置透视视窗的空间大小
    }
    public void onDrawFrame(GL10 gl) {
        //清除颜色缓存和深度缓存
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glMatrixMode(GL10.GL_MODELVIEW);                   //设置使用模型矩阵进行变换
        gl.glLoadIdentity();                                  //初始化单位矩阵
        //当使用 GL_MODELVIEW 模式时，必须设置视点，也就是观察点
        GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
        gl.glRotatef(1000, -0.1f, -0.1f, 0.05f);              //旋转总坐标系
        /*****************************************************************/
        /********************旋转*****************************************/
        long elapsed = System.currentTimeMillis() - startTime;   //计算逝去的时间
        gl.glRotatef(elapsed * (30f / 1000f), 0, 1, 0);       //在 Y 轴上旋转 30 度
        gl.glRotatef(elapsed * (15f / 1000f), 1, 0, 0);       //在 X 轴上旋转 15 度
        /********************************************************************/
        pyramid.draw(gl);                                     //绘制四棱锥
    }
}
```

15.4.2 使用 Android 机器人对立方体进行纹理贴图

【例 15.7】 本实例要求使用 OpenGL 技术绘制一个使用 Android 机器人进行纹理贴图的立方体，运行程序，效果如图 15.7 所示。



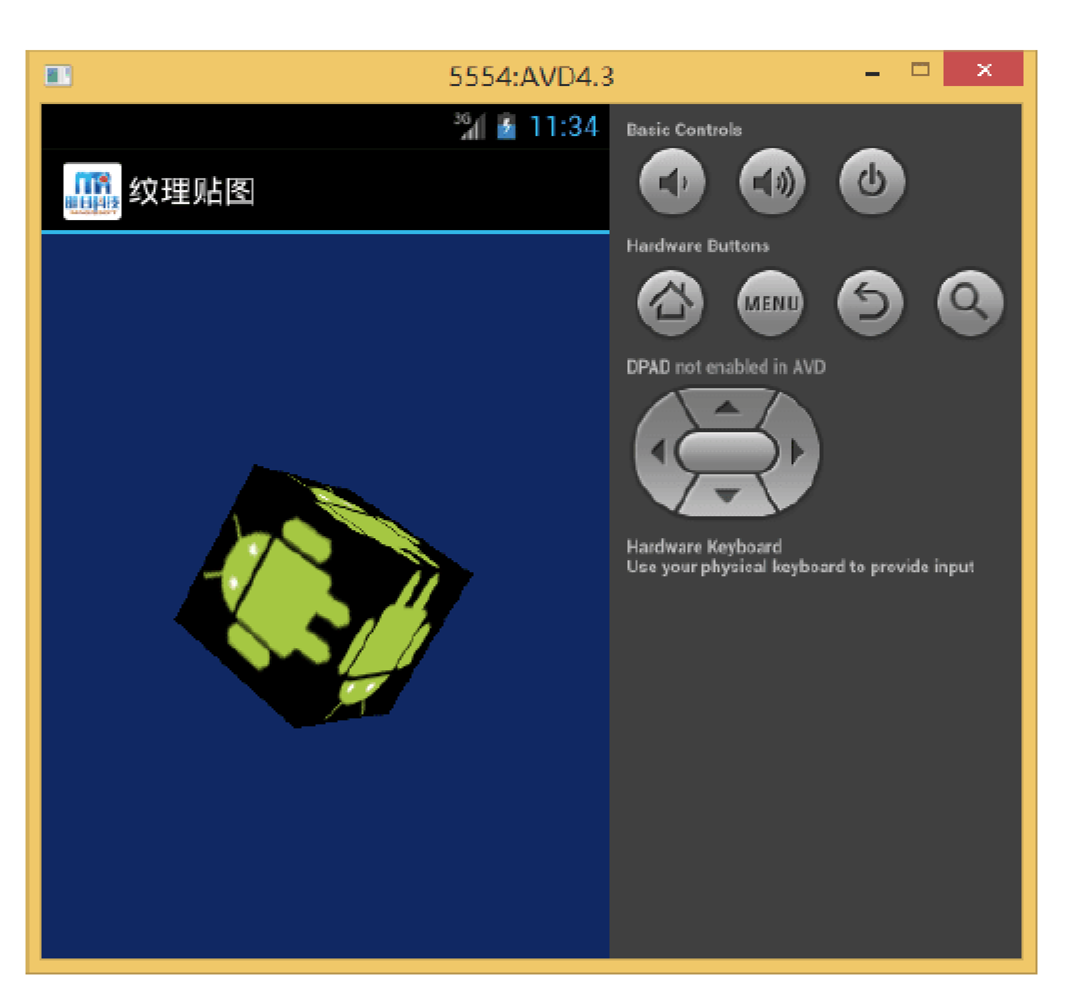


图 15.7　使用 Android 机器人对立方体进行纹理贴图

实例位置：光盘\MR\Instance\15\15.7

本实例是在例 15.2 的基础上实现的，与例 15.2 最大的不同的是，这里使用 Android 机器人对立方体进行纹理贴图。使用 Android 机器人对立方体进行纹理贴图的代码如下：

```
/**
 * 功能：进行纹理贴图
 */
void loadTexture(GL10 gl, Context context, int resource) {
    Bitmap bmp = BitmapFactory.decodeResource(context.getResources(),
        resource);                                              //加载位图
    GLUtils.texImage2D(GL10.GL_TEXTURE_2D, 0, bmp, 0);          //使用图片生成纹理
    bmp.recycle();                                              //释放资源
}
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.7f, 0.9f, 0.9f, 1.0f);                //设置窗体背景颜色
        gl.glClearColor(0.08f, 0.16f, 0.39f, 1.0f);             //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);           //启用顶点坐标数组
        gl.glDisable(GL10.GL_DITHER);                           //关闭抗抖动
        //设置系统对透视进行修正
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
        gl.glShadeModel(GL10.GL_SMOOTH);                        //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                        //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                         //设置深度测试的类型
        /************************应用纹理贴图******************************/
        gl.glEnableClientState(GL10.GL_TEXTURE_COORD_ARRAY);    //启用贴图坐标数组
        gl.glEnable(GL10.GL_TEXTURE_2D);                        //启用纹理贴图
        cube.loadTexture(gl, context, R.drawable.android);      //进行纹理贴图
        /****************************************************************/
    }
```





Note

15.5　本章常见错误

使用 Android OpenGL 绘制动画时，动画中出现了黑边，如何解决该问题呢？

如果使用的 png 图片是预乘（一种存储 alpha 的方法，表示一个图像用它自己的 RGB 通道乘以它自己的 ALPHA 通道）的，并且使用 gl.glBlendFunc(GL10.GL_SRC_ALPHA,GL10.GL_ONE_MINUS_SRC_ALPHA)就会产生黑边，解决该问题的方法是使用 gl.glBlendFunc(GL10.GL_ONE, GL10.GL_ONE_MINUS_SRC_ALPHA)。

15.6　本 章 小 结

本章首先简要介绍了 OpenGL 和 OpenGL ES，Android 系统内置了对 OpenGL ES 的支持，使用 OpenGL ES 可以开发出很好的 3D 产品，包括 3D 游戏，然后介绍了如何绘制 3D 图形和为 3D 图形添加纹理贴图、旋转效果、光照颜色和透明效果，这些内容都是进行 3D 产品开发的基础，希望读者重点掌握。

15.7　跟 我 上 机

参考答案：光盘\MR\跟我上机

开发一个 Android 程序，要求使用 OpenGL 技术绘制一个三棱锥。

三棱锥有 4 个面，而且每一个面都是由三角形组成的，这正好符合 OpenGL ES 的绘图机制，所有图形都是由三角形组成的。首先定义一个 GLTriPyramid 类，用来定义三棱锥的坐标点及绘制三棱锥的方法。其代码如下：

```
public class GLTriPyramid {
    private final IntBuffer mVertexBuffer;
    public GLTriPyramid() {
        int one = 65536;
        int half = one / 2;
        //三棱锥
        int vertices[] = {
                //LEFT
                0, half, 0, -half, -half, 0, half, -half, half,
                //RIGHT
                0, half, 0, -half, -half, 0, half, -half, 0,
                //BACK
                0, half, 0, half, -half, half, half, -half, 0,
```

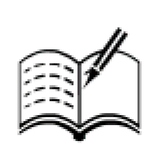


```
                //BOTTOM
                half, -half, 0, -half, -half, 0, half, -half, half, };
        ByteBuffer vbb = ByteBuffer.allocateDirect(vertices.length * 4);
        vbb.order(ByteOrder.nativeOrder());
        mVertexBuffer = vbb.asIntBuffer();
        mVertexBuffer.put(vertices);
        mVertexBuffer.position(0);
    }
    public void draw(GL10 gl) {
        gl.glVertexPointer(3, GL10.GL_FIXED, 0, mVertexBuffer);
        //绘制 Left 面
        gl.glColor4f(1, 0, 0, 0.5f);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 0, 3);
        //绘制 RIGHT 面
        gl.glColor4f(0, 1, 0, 0.5f);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 3, 3);
        //绘制 BACK 面
        gl.glColor4f(0, 0, 1, 0.5f);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 6, 3);
        //绘制 BOTTOM 面
        gl.glColor4f(0, 1, 1, 0.5f);
        gl.glDrawArrays(GL10.GL_TRIANGLE_STRIP, 9, 3);
    }
}
```

定义一个 TriPyramidRenderer 类，继承自 GLSurfaceView.Renderer 接口，然后实现其中的 onSurfaceCreated()、onDrawFrame()和 onSurfaceChanged()方法，在这 3 个方法中分别对三棱锥的背景颜色、场景等进行设置，从而绘制一个每个面颜色不同的三棱锥。其代码如下：

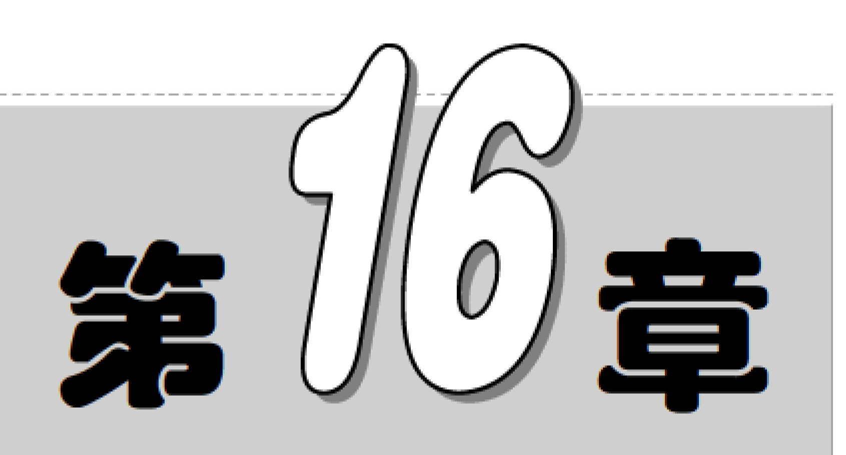
```
public class TriPyramidRenderer implements GLSurfaceView.Renderer {
    private final GLTriPyramid cube ;
    private long startTime;
    public TriPyramidRenderer(){
        cube = new GLTriPyramid();
        startTime=System.currentTimeMillis();
    }
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.5f, 0.5f, 0.5f, 1);                //设置窗体背景颜色
        gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
        gl.glDisable(GL10.GL_DITHER);                        //关闭抗抖动
        //设置系统对透视进行修正
        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT, GL10.GL_FASTEST);
        gl.glShadeModel(GL10.GL_SMOOTH);                     //设置阴影平滑模式
        gl.glEnable(GL10.GL_DEPTH_TEST);                     //启用深度测试
        gl.glDepthFunc(GL10.GL_LEQUAL);                      //设置深度测试的类型
    }
    public void onDrawFrame(GL10 gl) {
        //重绘背景颜色
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
```





Note

```
        /********* 将屏幕设置为黑色 *********************/
        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();                                //初始化单位矩阵
        /**********************************************/
        //当使用 GL_MODELVIEW 时，必须设置视窗的位置
        GLU.gluLookAt(gl, 0, 0, -5, 0f, 0f, 0f, 0f, 1.0f, 0.0f);
        gl.glRotatef(1000, -0.1f, -0.1f, 0.05f);            //旋转
        cube.draw(gl);                                      //绘制三棱锥
    }
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
        float ratio = (float) width / height;               //计算透视视窗的宽度、高度比
        gl.glMatrixMode(GL10.GL_PROJECTION);                //将当前矩阵模式设为投影矩阵
        gl.glLoadIdentity();                                //初始化单位矩阵
        GLU.gluPerspective(gl, 45.0f, ratio, 1, 100f);      //设置透视视窗的空间大小
    }
}
```



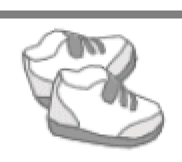
多媒体应用开发

（视频讲解：50 分钟）

随着 3G 时代的到来，在手机和平板电脑上应用多媒体已经非常广泛了。Android 作为又一大手机、平板电脑操作系统，对于多媒体应用也提供了良好的支持。它不仅支持音频和视频的播放，而且还支持录制音频和摄像头拍照等。本章将对 Android 中的音频及视频等多媒体应用进行详细介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- □ 实现包括播放、暂停/继续和停止功能的音乐播放器
- □ 使用 SoundPool 播放音频
- □ 使用 VideoView 组件播放视频
- □ 使用 MediaPlayer 和 SurfaceView 播放视频
- □ 为游戏界面添加背景音乐和按键音
- □ 制作开场动画
- □ 开发带音量控制的音乐播放器





Note

16.1　音频的播放

Android 提供了对常用音频和视频格式的支持，它所支持的音频格式包括 MP3（.mp3）、3GPP（.3gp）、Ogg（.ogg）和 WAVE（.ave）等，支持的视频格式包括 3GPP（.3gp）和 MPEG-4（.mp4）等。通过 Android API 提供的相关方法，可以实现音频与视频的播放。下面将分别介绍播放音频与视频的不同方法。

16.1.1　使用 MediaPlayer 播放音频

在 Android 中，提供了 MediaPlayer 类用来播放音频。使用 MediaPlayer 类播放音频比较简单，只需要创建该类的对象，并为其指定要播放的音频文件，然后再调用它的 start()方法就可以播放音频文件了。下面将详细介绍如何使用 MediaPlayer 播放音频文件。

1．创建 MediaPlayer 对象，并装载音频文件

创建 MediaPlayer 对象，并装载音频文件。可以使用该类提供的静态方法 create()来实现，也可通过它的无参构造方法来创建并实例化该类的对象来实现。

MediaPlayer 类的静态方法 create()常用的语法格式有以下两种。

☑　create(Context context, int resid)

该方法用于从资源 ID 所对应的资源文件中装载音频，并返回新创建的 MediaPlayer 对象。例如，要创建装载音频资源（res/raw/d.wav）的 MediaPlayer 对象，可以使用下面的代码：

```
MediaPlayer player=MediaPlayer.create(this, R.raw.d);
```

☑　create(Context context, Uri uri)

该方法用于根据指定的 URI 来装载音频，并返回新创建的 MediaPlayer 对象。例如，要创建装载了音频文件（URI 地址为 http://www.mingribook.com/sound/bg.mp3）的 MediaPlayer 对象，可以使用下面的代码：

```
MediaPlayer player=MediaPlayer.create(this, Uri.parse("http://www.mingribook.com/sound/bg.mp3"));
```



说明：

在访问网络中的资源时，要在 AndroidManifest.xml 文件中授予该程序访问网络的权限。其具体的授权代码如下：

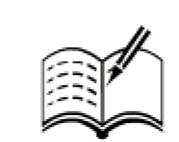
```
<uses-permission android:name="android.permission.INTERNET"/>
```

在通过 MediaPlayer 类的静态方法 create()来创建 MediaPlayer 对象时，已经装载了要播放的



音频，而使用无参的构造方法来创建 MediaPlayer 对象时，需要单独指定要装载的资源，这可以使用 MediaPlayer 类的 setDataSource()方法实现。

在使用 setDataSource()方法装载音频文件后，实际上 MediaPlayer 并未真正去装载该音频文件，还需要调用 MediaPlayer 的 prepare()方法去真正装载音频文件。使用无参的构造方法来创建 MediaPlayer 对象并装载指定的音频文件可以使用下面的代码：



Note

```
MediaPlayer player=new MediaPlayer();
try {
    player.setDataSource("/sdcard/s.wav");          //指定要装载的音频文件
} catch (IllegalArgumentException e1) {
    e1.printStackTrace();
} catch (SecurityException e1) {
    e1.printStackTrace();
} catch (IllegalStateException e1) {
    e1.printStackTrace();
} catch (IOException e1) {
    e1.printStackTrace();
}
    try {
            player.prepare();                        //预加载音频
    } catch (IllegalStateException e) {
            e.printStackTrace();
    } catch (IOException e) {
            e.printStackTrace();
      }
```

2．开始或恢复播放

在获取到 MediaPlayer 对象后，就可以使用 MediaPlayer 类提供的 start()方法来开始播放或恢复已经暂停的音频的播放。例如，已经创建了一个名称为 player，并且装载了要播放的音频，可以使用下面的代码播放该音频。

```
player.start();                                  //开始播放
```

3．停止播放

使用 MediaPlayer 类提供的 stop()方法可以停止正在播放的音频。例如，已经创建了一个名称为 player，并且已经开始播放装载的音频，可以使用下面的代码停止播放该音频。

```
player.stop();                                   //停止播放
```

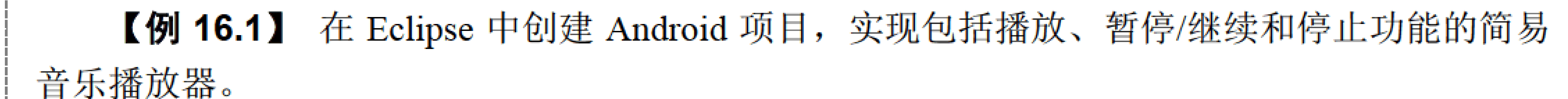
4．暂停播放

使用 MediaPlayer 类提供的 pause()方法可以暂停正在播放的音频。例如，已经创建了一个名称为 player，并且已经开始播放装载的音频，可以使用下面的代码暂停播放该音频。



```
player.pause();                                          //暂停播放
```

【例 16.1】 在 Eclipse 中创建 Android 项目，实现包括播放、暂停/继续和停止功能的简易音乐播放器。



实例位置：光盘\MR\Instance\16\16.1

程序的开发步骤如下：

（1）将要播放的音频文件上传到 SD 卡的根目录中，这里要播放的音频文件为 ninan.mp3。

（2）修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认添加的线性布局管理器中添加一个水平线性布局管理器，并在其中添加 3 个按钮，分别为“播放”、“暂停/继续”和“停止”按钮。

（3）打开默认添加的 MainActivity，在该类中，定义所需的成员变量。其具体代码如下：

```
private MediaPlayer player;                              //MediaPlayer 对象
private boolean isPause = false;                         //是否暂停
private File file;                                       //要播放的音频文件
private TextView hint;                                   //声明显示提示信息的文本框
```

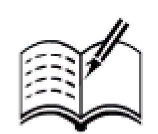
（4）在 onCreate()方法中，首先获取布局管理器中添加的“播放”按钮、“暂停/继续”按钮、“停止”按钮和显示提示信息的文本框，然后获取要播放的文件，最后再判断该文件是否存在，如果存在，则创建一个装载该文件的 MediaPlayer 对象，否则，显示提示信息，并设置“播放”按钮不可用。其关键代码如下：

```
final Button button1 = (Button) findViewById(R.id.button1);    //获取播放按钮
final Button button2 = (Button) findViewById(R.id.button2);    //获取“暂停/继续”按钮
final Button button3 = (Button) findViewById(R.id.button3);    //获取“停止”按钮
hint = (TextView) findViewById(R.id.hint);                     //获取用户显示提示信息的文本框
file = new File("/sdcard/ninan.mp3");                          //获取要播放的文件
if (file.exists()) {                                           //如果文件存在
    player = MediaPlayer
            .create(this, Uri.parse(file.getAbsolutePath())); //创建 MediaPlayer 对象
} else {
    hint.setText("要播放的音频文件不存在！");
    button1.setEnabled(false);
    return;
}
```

（5）编写用于播放音乐的 play()方法，该方法没有入口参数的返回值。在该方法中，首先调用 MediaPlayer 对象的 reset()方法重置 MediaPlayer 对象，然后重新为其设置要播放的音频文件，并预加载该音频，最后调用 start()方法开始播放音频，并修改显示提示信息的文本框中的内容。其具体代码如下：

```
private void play() {
    try {
```






```
            player.reset();
            player.setDataSource(file.getAbsolutePath());          //重新设置要播放的音频
            player.prepare();                                      //预加载音频
            player.start();                                        //开始播放
            hint.setText("正在播放音频...");
        } catch (Exception e) {
            e.printStackTrace();                                   //输出异常信息
        }
}
```

（6）为 MediaPlayer 对象添加完成事件监听器，用于当音乐播放完毕后，重新开始播放音乐。其具体代码如下：

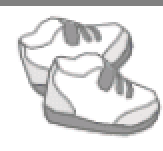
```
player.setOnCompletionListener(new OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        play();                                                    //重新开始播放
    }
});
```

（7）为“播放”按钮添加单击事件监听器，在重写的 onClick()方法中，首先调用 play()方法开始播放音乐，然后对代表是否暂停的标记变量 isPause 进行设置，最后再设置各按钮的可用状态。其关键代码如下：

```
button1.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        play();                                                    //开始播放音乐
        if (isPause) {
            button2.setText("暂停");
            isPause = false;                                       //设置暂停标记变量的值为 false
        }
        button2.setEnabled(true);                                  // “暂停/继续”按钮可用
        button3.setEnabled(true);                                  // “停止”按钮可用
        button1.setEnabled(false);                                 // “播放”按钮不可用
    }
});
```

（8）为“暂停/继续”按钮添加单击事件监听器，在重写的 onClick()方法中，如果 MediaPlayer 处于播放状态并且标记变量 isPause 的值为 false，则暂停播放音频，并设置相关信息，否则调用 MediaPlayer 对象的 start()方法继续播放音乐，并设置相关信息。其关键代码如下：

```
button2.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        if (player.isPlaying() && !isPause) {
```

```
                player.pause();                          //暂停播放
                isPause = true;
                ((Button) v).setText("继续");
                hint.setText("暂停播放音频...");
                button1.setEnabled(true);                //“播放”按钮可用
            } else {
                player.start();                          //继续播放
                ((Button) v).setText("暂停");
                hint.setText("继续播放音频...");
                isPause = false;
                button1.setEnabled(false);               //“播放”按钮不可用
            }
        }
});
```

（9）为“停止”按钮添加单击事件监听器，在重写的 onClick()方法中，首先调用 MediaPlayer 对象的 stop()方法停止播放音频，然后设置提示信息及各按钮的可用状态。其具体代码如下：

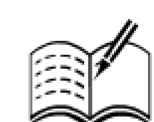
```
button3.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            player.stop();                               //停止播放
            hint.setText("停止播放音频...");
            button2.setEnabled(false);                   //“暂停/继续”按钮不可用
            button3.setEnabled(false);                   //“停止”按钮不可用
            button1.setEnabled(true);                    //“播放”按钮可用
        }
});
```

（10）重写 Acitivity 的 onDestroy()方法，用于在当前 Activity 销毁时，停止正在播放的视频，并释放 MediaPlayer 所占用的资源。其具体代码如下：

```
@Override
protected void onDestroy() {
        if(player.isPlaying()){
            player.stop();                               //停止音频的播放
        }
        player.release();                               //释放资源
        super.onDestroy();
}
```

运行本实例，显示一个简易音乐播放器，单击“播放”按钮，开始播放音乐，同时“播放”按钮变为不可用状态，而“暂停”按钮和“停止”按钮变为可用状态，如图 16.1 所示；单击“暂停”按钮，将暂停音乐的播放，同时“播放”按钮变为可用；单击“继续”按钮，将继续音乐的播放，同时“继续”按钮变为“暂停”按钮；单击“停止”按钮，将停止音乐的播放，同时“暂停/继续”和“停止”按钮变为不可用，“播放”按钮可用。





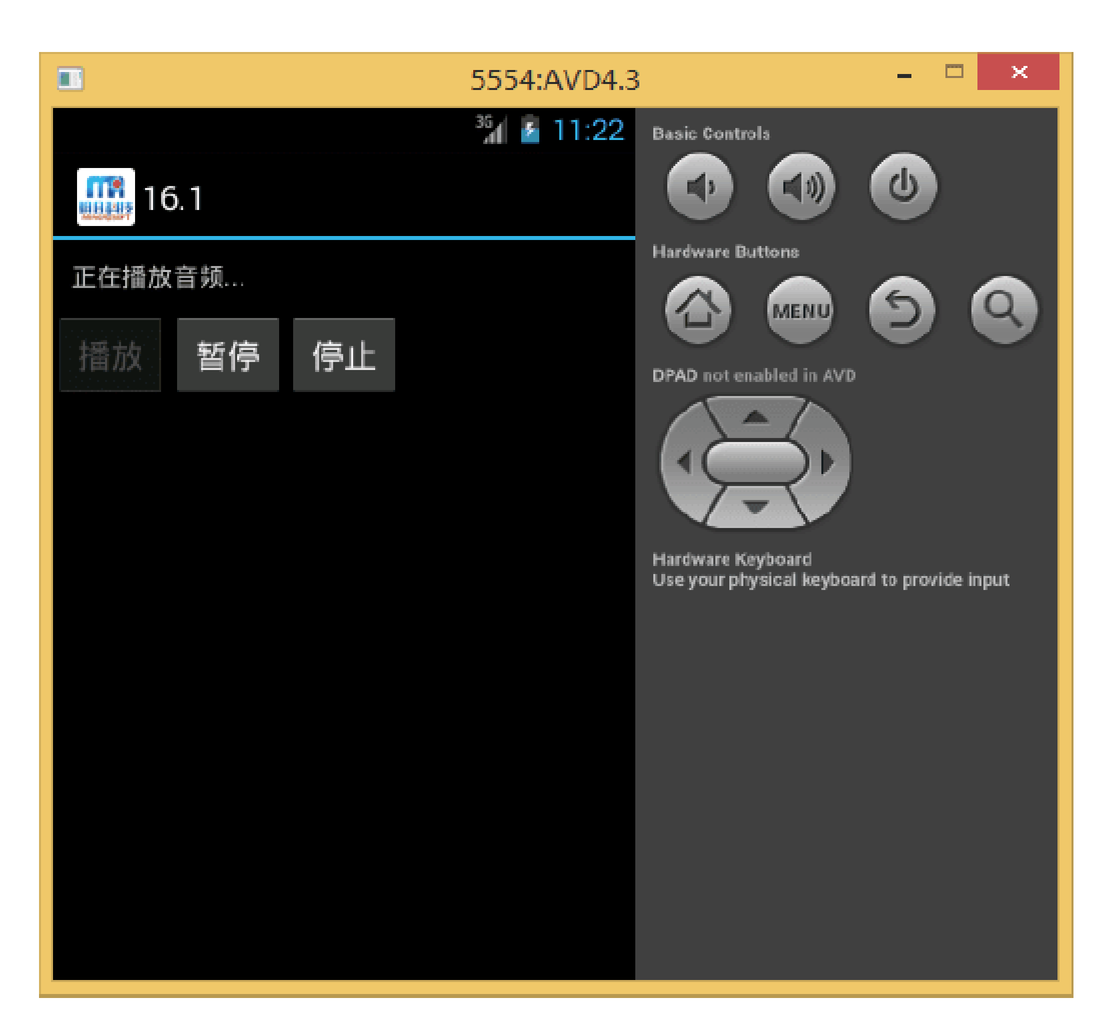


图 16.1　简易音乐播放器

16.1.2　使用 SoundPool 播放音频

由于 MediaPlayer 占用资源较高，且不支持同时播放多个音频，所以 Android 还提供了另一个播放音频的 SoundPool。SoundPool 也就是音频池，它可以同时播放多个短促的音频，而且占用的资源少。SoundPool 适合在应用程序中播放按键音或者消息提示音等，在游戏中实现密集而短暂的声音，如多个飞机的爆炸声等。使用 SoundPool 播放音频，首先需要创建 SoundPool 对象，然后加载所要播放的音频，最后再调用 play()方法播放音频。下面进行详细介绍。

1．创建 SoundPool 对象

SoundPool 类提供了一个构造方法，用来创建 SoundPool 对象。该构造方法的语法格式如下：

```
SoundPool(int maxStreams, int streamType, int srcQuality)
```

其中，参数 maxStreams 用于指定可以容纳多少个音频；参数 streamType 用于指定声音类型，可以通过 AudioManager 类提供的常量进行指定，通常使用 STREAM_MUSIC；参数 srcQuality 用于指定音频的品质，0 为默认值。

例如，创建一个可以容纳 10 个音频的 SoundPool 对象，可以使用以下代码。

```
SoundPool soundpool = new SoundPool(10,
        AudioManager.STREAM_SYSTEM, 0);    //创建一个 SoundPool 对象，该对象可以容纳 10 个音频流
```

2．加载所要播放的音频

创建 SoundPool 对象后，可以调用它的 load()方法来加载要播放的音频。load()方法的语法格式有以下 4 种。

☑　public int load(Context context, int resId, int priority)

该方法用于通过指定的资源 ID 来加载音频。




Note

☑ public int load(String path, int priority)

该方法用于通过音频文件的路径来加载音频。

☑ public int load(AssetFileDescriptor afd, int priority)

该方法用于从 AssetFileDescriptor 所对应的文件中加载音频。

☑ public int load(FileDescriptor fd, long offset, long length, int priority)

该方法用于加载 FileDescriptor 对象中，从 offset 开始，长度为 length 的音频。

例如，要通过资源 ID 来加载音频文件 ding.wav，可以使用下面的代码。

```
soundpool.load(this, R.raw.ding, 1);
```



说明：

为了更好地管理所加载的每个音频，一般使用 HashMap<Integer, Integer>对象来管理这些音频。这时可以先创建一个 HashMap<Integer, Integer>对象，然后应用该对象的 put()方法将加载的音频保存到该对象中。例如，创建一个 HashMap<Integer, Integer>对象，并应用 put()方法添加一个音频可以使用下面的代码。

```
HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();  //创建一个 HashMap 对象
soundmap.put(1, soundpool.load(this, R.raw.chimes, 1));
```

3．播放音频

调用 SoundPool 对象的 play()方法可播放指定音频。play()方法的语法格式如下：

```
play(int soundID, float leftVolume, float rightVolume, int priority, int loop, float rate)
```

play()方法的各参数说明如表 16.1 所示。

表 16.1　play()方法的参数说明

参　数	说　明
soundID	用于指定要播放的音频，该音频为通过 load()方法返回的音频
leftVolume	用于指定左声道的音量，取值范例为 0.0～1.0
rightVolume	用于指定右声道的音量，取值范例为 0.0～1.0
priority	用于指定播放音频的优先级，数值越大，优先级越高
loop	用于指定循环次数，0 为不循环，-1 为循环
rate	用于指定速率，1 为正常，最低为 0.5，最高为 2

例如，要播放音频资源中保存的音频文件 notify.wav，可以使用下面的代码。

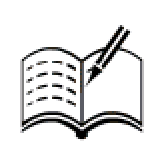
```
soundpool.play(soundpool.load(MainActivity.this, R.raw.notify, 1), 1, 1, 0, 0, 1);          //播放指定的音频
```

【例 16.2】 在 Eclipse 中创建 Android 项目，实现通过 SoundPool 播放音频。

 **实例位置：光盘\MR\Instance\16\16.2**

程序的开发步骤如下：





（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加 4 个按钮，分别为“风铃声”、“布谷鸟叫声”、“门铃声”和“电话声”按钮。

（2）打开默认添加的 MainActivity，在该类中，创建两个成员变量。其具体代码如下：

```
private SoundPool soundpool;                                    //声明一个 SoundPool 对象
//创建一个 HashMap 对象
private HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();
```

（3）在 onCreate()方法中，首先获取布局管理器中添加的“风铃声”、“布谷鸟叫声”、“门铃声”和“电话声”按钮，然后实例化 SoundPool 对象，再将要播放的全部音频流保存到 HashMap 对象中。其具体代码如下：

```
Button chimes = (Button) findViewById(R.id.button1);          //获取“风铃声”按钮
Button enter = (Button) findViewById(R.id.button2);           //获取“布谷鸟叫声”按钮
Button notify = (Button) findViewById(R.id.button3);          //获取“门铃声”按钮
Button ringout = (Button) findViewById(R.id.button4);         //获取“电话声”按钮
soundpool = new SoundPool(5,
        AudioManager.STREAM_SYSTEM, 0);//创建一个 SoundPool 对象，该对象可以容纳 5 个音频流
//将要播放的音频流保存到 HashMap 对象中
soundmap.put(1, soundpool.load(this, R.raw.chimes, 1));
soundmap.put(2, soundpool.load(this, R.raw.enter, 1));
soundmap.put(3, soundpool.load(this, R.raw.notify, 1));
soundmap.put(4, soundpool.load(this, R.raw.ringout, 1));
soundmap.put(5, soundpool.load(this, R.raw.ding, 1));
```

（4）分别为“风铃声”、“布谷鸟叫声”、“门铃声”和“电话声”按钮添加单击事件监听器，在重写的 onClick()方法中播放指定音频。其具体代码如下：

```
chimes.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        soundpool.play(soundmap.get(1), 1, 1, 0, 0, 1);         //播放指定的音频
    }
});
enter.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        soundpool.play(soundmap.get(2), 1, 1, 0, 0, 1);         //播放指定的音频
    }
});
notify.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        soundpool.play(soundmap.get(3), 1, 1, 0, 0, 1);         //播放指定的音频
    }
});
ringout.setOnClickListener(new OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
        soundpool.play(soundmap.get(4), 1, 1, 0, 0, 1);          //播放指定的音频
        }
});
```

Note

（5）重写键盘按键被按下的方法 onKeyDown()，用于实现播放按键音的功能。其具体代码如下：

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
        soundpool.play(soundmap.get(5), 1, 1, 0, 0, 1);          //播放按键音
        return true;
}
```

运行本实例，将显示如图 16.2 所示的运行效果。单击“风铃声”、“布谷鸟叫声”等按钮，播放相应的音乐；按下键盘上的按钮，播放一个按键音。

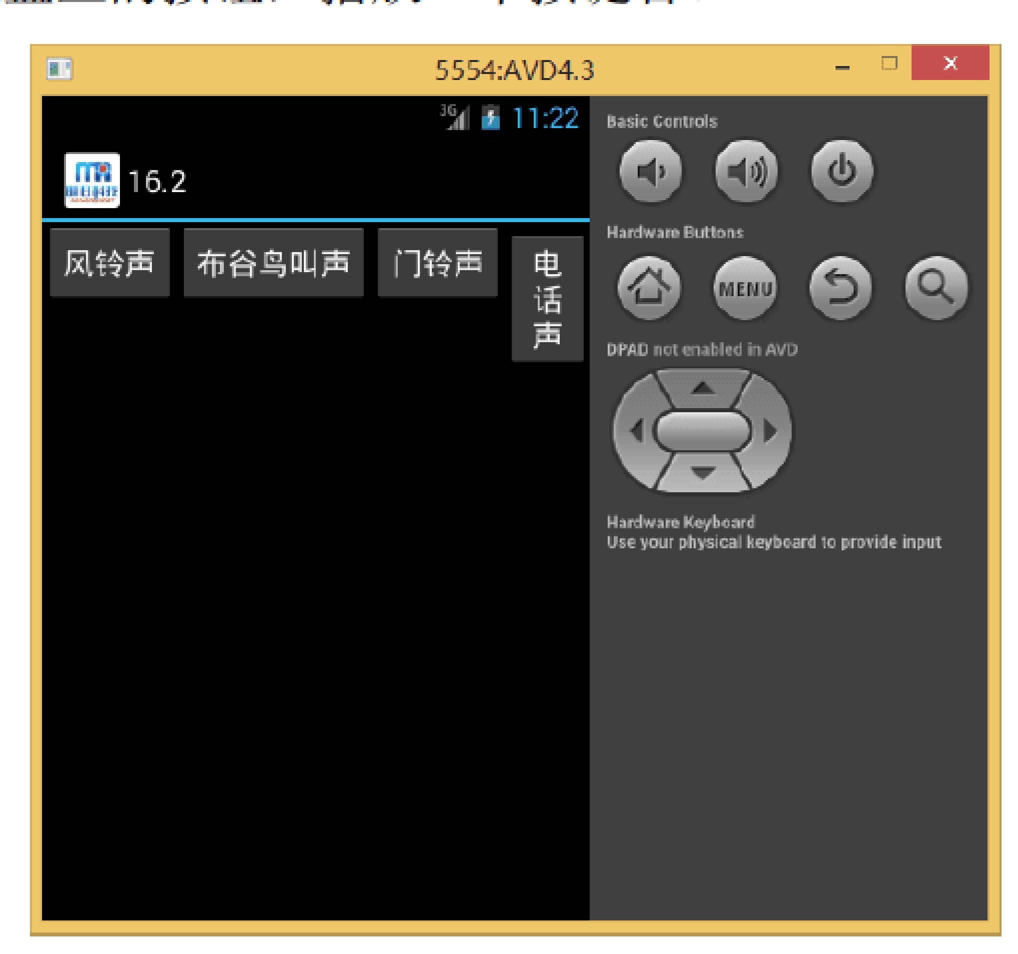


图 16.2　应用 SoundPool 播放音频

16.2　视频的播放

16.2.1　使用 VideoView 组件播放视频

在 Android 中提供了一个 VideoView 组件，用于播放视频文件。要想使用 VideoView 组件播放视频，首先需要在布局文件中创建该组件，然后在 Activity 中获取该组件，并应用其 setVideoPath()或 setVideoURI()方法加载要播放的视频，最后调用 VideoView 组件的 start()方法来播放视频。另外，VideoView 组件还提供了 stop()和 pause()方法来停止或暂停视频的播放。

在布局文件中创建 VideoView 组件的基本语法格式如下：



```
<VideoView
    属性列表
</VideoView>
```

VideoView 组件支持的 XML 属性如表 16.2 所示。



表 16.2　VideoView 组件支持的 XML 属性

XML 属性	描　　述
android:id	用于设置组件的 id
android:background	用于设置背景，可以设置背景图片，也可以设置背景颜色
android:layout_gravity	用于设置对齐方式
android:layout_width	用于设置宽度
android:layout_height	用于设置高度

在 Android 中还提供了一个可以与 VideoView 组件结合使用的 MediaController 组件，用于通过图形控制界面来控制视频的播放。

下面通过一个具体的实例来说明如何使用 VideoView 和 MediaController 组件来播放视频。

【例 16.3】 在 Eclipse 中创建 Android 项目，实现通过 VideoView 组件播放视频。

实例位置：光盘\MR\Instance\16\16.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加一个 VideoView 组件用于播放视频文件。其关键代码如下：

```
<VideoView
    android:id="@+id/video"
    android:background="@drawable/mpbackground"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center" />
```

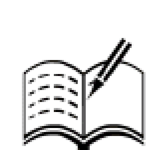
（2）打开默认添加的 MainActivity，在该类中声明一个 VideoView 对象。其具体代码如下：

```
private VideoView video;                              //声明 VideoView 对象
```

（3）在 onCreate()方法中，首先获取布局管理器中添加的 VideoView，并创建一个要播放视频所对应的 File 对象，然后创建一个 MediaController 对象，用于控制视频的播放，最后再判断要播放的视频文件是否存在，如果存在，使用 VideoView 播放该视频，否则，显示消息提示框显示提示信息。其具体代码如下：

```
video=(VideoView) findViewById(R.id.video);           //获取 VideoView 组件
File file=new File("/sdcard/mingrisoft.mp4");         //获取 SD 卡上要播放的文件
MediaController mc=new MediaController(MainActivity.this);
if(file.exists()){                                    //判断要播放的视频文件是否存在
```





Note

```
        video.setVideoPath(file.getAbsolutePath());          //指定要播放的视频
        video.setMediaController(mc);                        //设置 VideoView 与 MediaController 相关联
        video.requestFocus();                                //让 VideoView 获得焦点
        try {
            video.start();                                   //开始播放视频
        } catch (Exception e) {
            e.printStackTrace();                             //输出异常信息
        }
        //为 VideoView 添加完成事件监听器
        video.setOnCompletionListener(new OnCompletionListener() {
        @Override
                public void onCompletion(MediaPlayer mp) {
                Toast.makeText(MainActivity.this, "视频播放完毕！ ", Toast.LENGTH_SHORT).show();
                }
            });
        }else{
            Toast.makeText(this, "要播放的视频文件不存在", Toast.LENGTH_SHORT).show();
}
```

实例运行效果如图 16.3 所示。

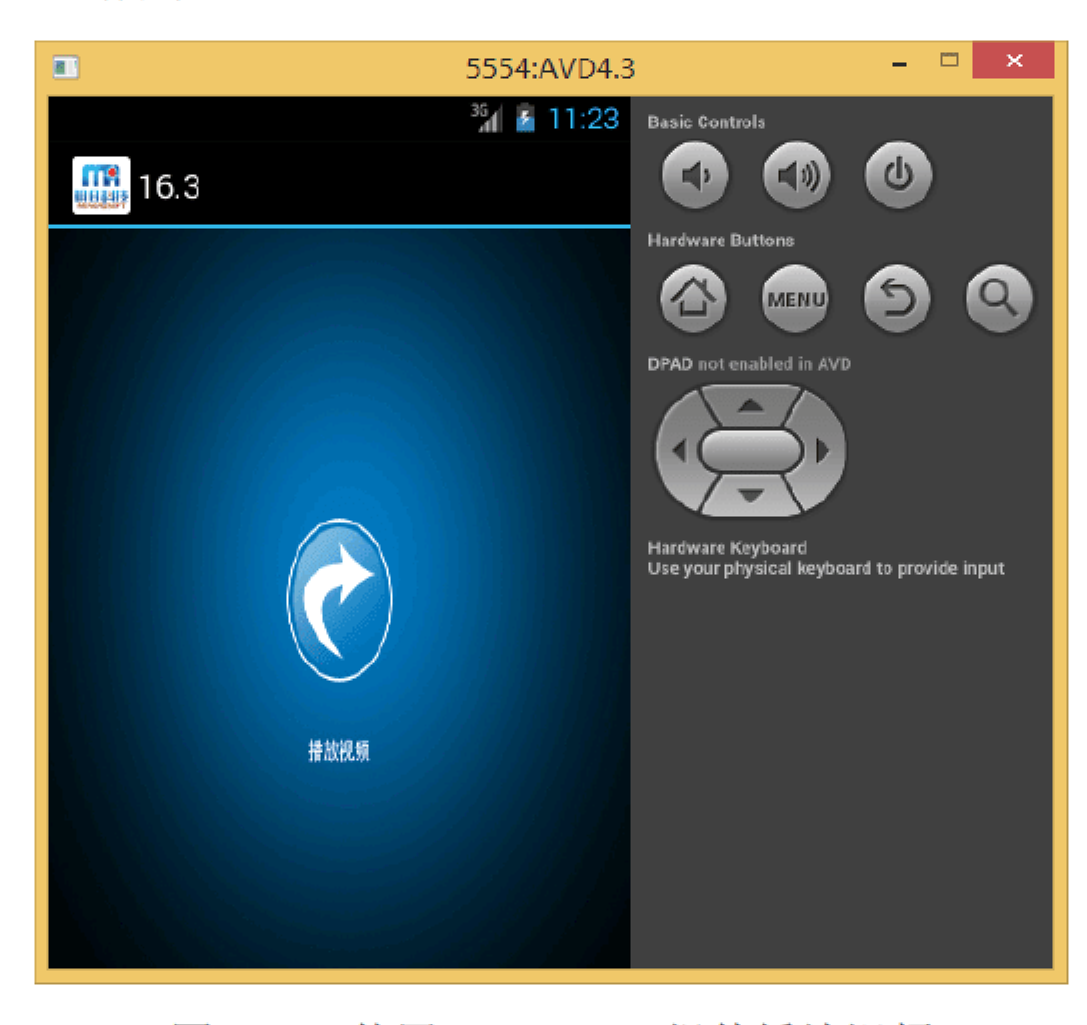


图 16.3　使用 VideoView 组件播放视频

16.2.2　使用 MediaPlayer 和 SurfaceView 播放视频

16.1.1 节介绍了使用 MediaPlayer 播放音频，实际上，MediaPlayer 还可以用来播放视频文件，只不过使用 MediaPlayer 播放视频时，没有提供图像输出界面。这时，可以使用 SurfaceView 组件来显示视频图像。使用 MediaPlayer 和 SurfaceView 来播放视频，大致可以分为以下 4 个步骤。

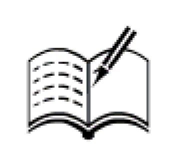
1．定义 SurfaceView 组件

定义 SurfaceView 组件可以在布局管理器中实现，也可以直接在 Java 代码中创建，不过推荐



使用在布局管理器中创建。在布局管理器中定义 SurfaceView 组件的基本语法格式如下：

```
<SurfaceView
    android:id="@+id/ID 号"
    android:background="背景"
    android:keepScreenOn="true|false"
    android:layout_width="宽度"
    android:layout_height="高度"/>
```



在上面的语法中，android:keepScreenOn 属性用于指定在播放视频时，是否打开屏幕。

例如，在布局管理器中，添加一个 id 号为 surfaceView1 的，设置了背景的 SurfaceView 组件，可以使用下面的代码。

```
<SurfaceView
    android:id="@+id/surfaceView1"
    android:background="@drawable/bg"
    android:keepScreenOn="true"
    android:layout_width="576px"
    android:layout_height="432px"/>
```

2．创建 MediaPlayer 对象，并为其加载要播放的视频

与播放音频时创建 MediaPlayer 对象一样，也可以使用 MediaPlayer 类的静态方法 create()和无参的构造方法创建 MediaPlayer 对象。

3．将所播放的视频画面输出到 SurfaceView

使用 MediaPlayer 对象的 setDisplay()方法可以将所播放的视频画面输出到 SurfaceView。setDisplay()方法的语法格式如下：

```
setDisplay(SurfaceHolder sh)
```

参数 sh 用于指定 SurfaceHolder 对象，可以通过 SurfaceView 对象的 getHolder()方法获得。

例如，为 MediaPlayer 对象指定输出视频画面的 SurfaceView，可以使用下面的代码。

```
mediaplayer.setDisplay(surfaceview.getHolder());    //设置将视频画面输出到 SurfaceView
```

4．调用 MediaPlayer 对象的相应方法控制视频的播放

使用 MediaPlayer 对象提供的 play()、pause()和 stop()方法，可以控制视频的播放、暂停和停止。

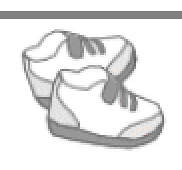
下面通过一个具体的实例来演示如何使用 MediaPlayer 和 SurfaceView 来播放视频。

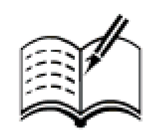
【例 16.4】 在 Eclipse 中创建 Android 项目，实现通过 MeidaPlayer 和 SurfaceView 播放视频。

实例位置：光盘\MR\Instance\16\16.4

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件





删除，然后在默认添加的线性布局管理器中添加一个用于显示视频图像的 SurfaceView 组件和一个水平线性布局管理器，在该水平线性布局管理器中，添加 3 个按钮，分别为“播放”、“暂停/继续”和“停止”按钮。其关键代码如下：

```
<SurfaceView
    android:id="@+id/surfaceView1"
    android:layout_width="264dp"
    android:layout_height="234dp"
    android:background="@drawable/bg"
    android:keepScreenOn="true" />
```

（2）打开默认添加的 MainActivity，在该类中声明一个 MediaPlayer 对象和一个 SurfaceView 对象。其具体代码如下：

```
private MediaPlayer mp;                              //声明 MediaPlayer 对象
private SurfaceView sv;                              //声明 SurfaceView 对象
```

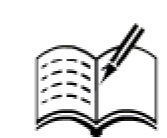
（3）在 onCreate()方法中，首先实例化 MediaPlayer 对象，然后获取布局管理器中添加的 SurfaceView 组件，最后再分别获取“播放”、“暂停/继续”和“停止”按钮。其具体代码如下：

```
mp=new MediaPlayer();                                //实例化 MediaPlayer 对象
sv=(SurfaceView)findViewById(R.id.surfaceView1);     //获取布局管理器中添加的 SurfaceView 组件
Button play=(Button)findViewById(R.id.play);         //获取“播放”按钮
final Button pause=(Button)findViewById(R.id.pause); //获取“暂停/继续”按钮
Button stop=(Button)findViewById(R.id.stop);         //获取“停止”按钮
```

（4）分别为“播放”、“暂停/继续”和“停止”按钮添加单击事件监听器，并在重写的 onClick()方法中实现播放视频、暂停/继续播放视频和停止播放视频等功能。其具体代码如下：

```
//为“播放”按钮添加单击事件监听器
 play.setOnClickListener(new OnClickListener() {
       @Override
       public void onClick(View v) {
              mp.reset();                                    //重置 MediaPlayer 对象
              try {
                     mp.setDataSource("/sdcard/ccc.mp4");    //设置要播放的视频
                     mp.setDisplay(sv.getHolder());          //设置将视频画面输出到 SurfaceView
                     mp.prepare();                           //预加载视频
                     mp.start();                             //开始播放
                     sv.setBackgroundResource(R.drawable.bg_playing);//改变 SurfaceView 的背景图片
                     pause.setText("暂停");
                     pause.setEnabled(true);                 //设置“暂停”按钮可用
              } catch (IllegalArgumentException e) {
                     e.printStackTrace();
              } catch (SecurityException e) {
                     e.printStackTrace();
              } catch (IllegalStateException e) {
                     e.printStackTrace();
              } catch (IOException e) {
```





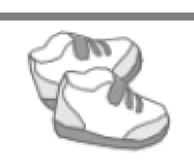

```
                e.printStackTrace();
            }
        }
});
//为“停止”按钮添加单击事件监听器
stop.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if(mp.isPlaying()){
                mp.stop();                                          //停止播放
                sv.setBackgroundResource(R.drawable.bg_finish);     //改变 SurfaceView 的背景图片
                pause.setEnabled(false);                            //设置“暂停”按钮不可用
            }
        }
});
//为“暂停”按钮添加单击事件监听器
pause.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if(mp.isPlaying()){
                mp.pause();                                         //暂停视频的播放
                ((Button)v).setText("继续");
            }else{
                mp.start();                                         //继续视频的播放
                ((Button)v).setText("暂停");
            }
        }
});
```

（5）为 MediaPlayer 对象添加完成事件监听器，在重写的 onCompletion()方法中改变 SurfaceView 的背景图片并弹出消息提示框显示视频已经播放完毕。其具体代码如下：

```
mp.setOnCompletionListener(new OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            sv.setBackgroundResource(R.drawable.bg_finish);         //改变 SurfaceView 的背景图片
            Toast.makeText(MainActivity.this, "视频播放完毕！", Toast.LENGTH_SHORT).show();
        }
});
```

（6）重写 Acitivity 的 onDestroy()方法，用于在当前 Activity 销毁时停止正在播放的视频，并释放 MediaPlayer 所占用的资源。其具体代码如下：

```
@Override
protected void onDestroy() {
        if(mp.isPlaying()){
            mp.stop();                                              //停止播放视频
        }
        mp.release();                                               //释放资源
```



```
        super.onDestroy();
}
```

Note

运行本实例，如图 16.4 所示，单击“播放”按钮，开始播放视频，并且让“暂停”按钮可用；单击“暂停”按钮，暂停视频的播放，同时该按钮变为“继续”按钮；单击“停止”按钮，停止正在播放的视频。

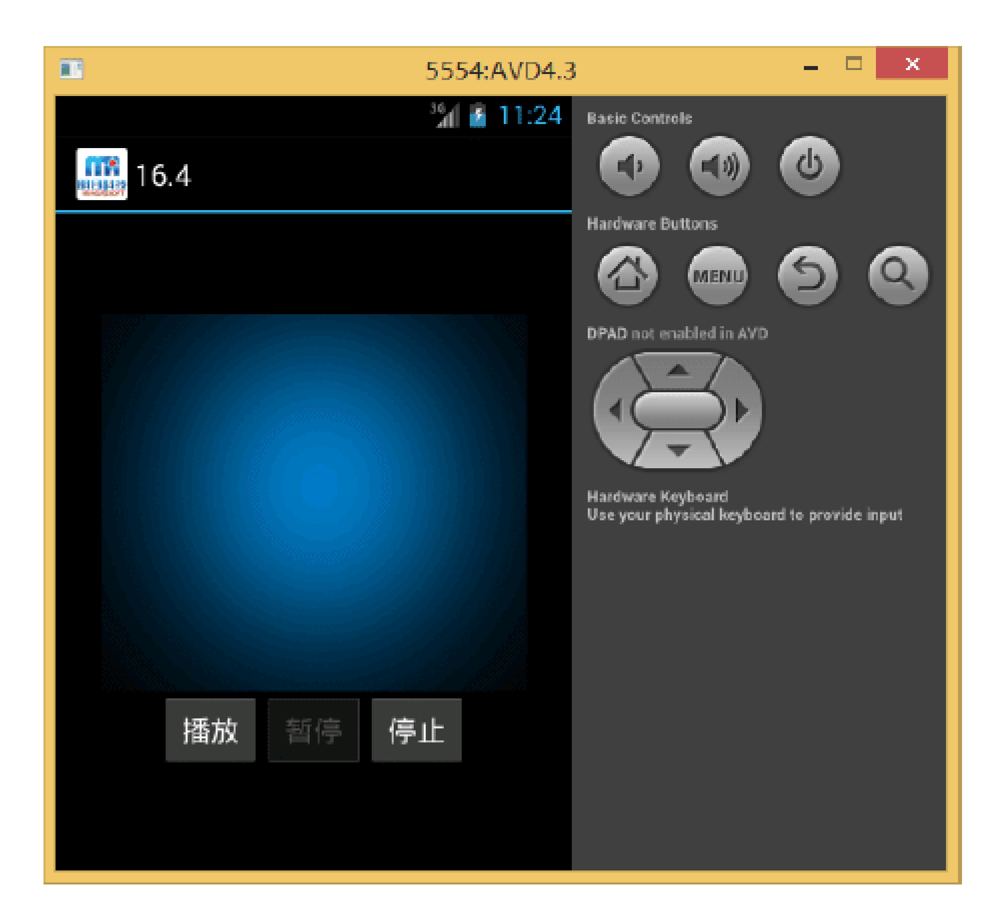


图 16.4　使用 MediaPlayer 和 SurfaceView 播放视频

16.3　综合应用

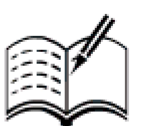
16.3.1　为游戏界面添加背景音乐和按键音

【例 16.5】 本实例主要实现为游戏界面添加背景音乐和按键音的功能，实例的运行效果如图 16.5 所示。



图 16.5　为游戏界面添加背景音乐和按键音





实例位置：光盘\MR\Instance\16\16.5

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个 FrameLayout 帧布局管理器，并在该布局管理器中添加一个 ImageView，用于显示一只小兔子，另外，还需要为添加的帧布局管理器设置背景图片。

（2）打开默认添加的 MainActivity，在该类中创建程序中所需的成员变量。其具体代码如下：

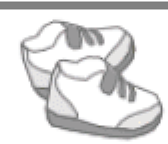
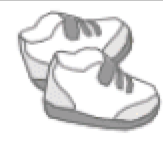
```
private SoundPool soundpool;                                    //声明一个 SoundPool 对象
private HashMap<Integer, Integer> soundmap = new HashMap<Integer, Integer>();
//创建一个 HashMap 对象
private ImageView rabbit;
private int x=0;                                                //兔子在 X 轴的位置
private int y=0;                                                //兔子在 Y 轴的位置
private int width=0;                                            //屏幕的宽度
private int height=0;                                           //屏幕的高度
```

（3）在 onCreate()方法中，首先实例化 SoundPool 对象，并将要播放的全部音频流保存到 HashMap 对象中，然后获取布局管理器中添加的小兔子，并获取屏幕的宽度和高度，再计算小兔子在 X 轴和 Y 轴的位置，最后通过 setX()和 setY()方法设置兔子的默认位置。其具体代码如下：

```
soundpool = new SoundPool(5, AudioManager.STREAM_SYSTEM, 0);
//创建一个 SoundPool 对象，该对象可以容纳 5 个音频流
//将要播放的音频流保存到 HashMap 对象中
soundmap.put(1, soundpool.load(this, R.raw.chimes, 1));
soundmap.put(2, soundpool.load(this, R.raw.enter, 1));
soundmap.put(3, soundpool.load(this, R.raw.notify, 1));
soundmap.put(4, soundpool.load(this, R.raw.ringout, 1));
soundmap.put(5, soundpool.load(this, R.raw.ding, 1));
rabbit=(ImageView)findViewById(R.id.rabbit);
width= MainActivity.this.getResources().getDisplayMetrics().widthPixels;
height=MainActivity.this.getResources().getDisplayMetrics().heightPixels;
x=width/2-44;                                                   //计算兔子在 X 轴的位置
y=height/2-35;                                                  //计算兔子在 Y 轴的位置
rabbit.setX(x);                                                 //设置兔子在 X 轴的位置
rabbit.setY(y);                                                 //设置兔子在 Y 轴的位置
```

（4）重写键盘的按键被按下的 onKeyDown()方法，在该方法中，应用 switch()语句分别为上、下、左、右方向键和其他按键指定不同的按键音，同时，在按下上、下、左和右方向键时，还会控制小兔子在相应方向上移动。其具体代码如下：

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch(keyCode){
        case KeyEvent.KEYCODE_DPAD_LEFT:                //向左方向键
            soundpool.play(soundmap.get(1), 1, 1, 0, 0, 1); //播放指定的音频
```

Note

```
            if(x>0){
                x-=10;
                rabbit.setX(x);                               //移动小兔子
            }
            break;
        case KeyEvent.KEYCODE_DPAD_RIGHT:                     //向右方向键
            soundpool.play(soundmap.get(2), 1, 1, 0, 0, 1);   //播放指定的音频
            if(x<width-88){
                x+=10;
                rabbit.setX(x);                               //移动小兔子
            }
            break;
        case KeyEvent.KEYCODE_DPAD_UP:                        //向上方向键
            soundpool.play(soundmap.get(3), 1, 1, 0, 0, 1);   //播放指定的音频
            if(y>0){
                y-=10;
                rabbit.setY(y);                               //移动小兔子
            }
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN:                      //向下方向键
            soundpool.play(soundmap.get(4), 1, 1, 0, 0, 1);   //播放指定的音频
            if(y<height-70){
                y+=10;
                rabbit.setY(y);                               //移动小兔子
            }
            break;
        default:
            soundpool.play(soundmap.get(5), 1, 1, 0, 0, 1);   //播放默认按键音
    }
    return super.onKeyDown(keyCode, event);
}
```

（5）在 res 目录下创建一个 menu 子目录，并在该目录中创建一个名称为 setting.xml 的菜单资源，在该文件中添加一个控制是否播放背景音乐的多选菜单组，默认为选中状态。setting.xml 文件的具体代码如下：

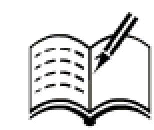
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <group android:id="@+id/setting" android:checkableBehavior="all">
    <item android:id="@+id/bgsound" android:title="播放背景音乐" android:checked="true"></item>
    </group>
</menu>
```

（6）重写 onCreateOptionsMenu()方法，应用步骤（5）中添加的菜单文件，创建一个选项菜单，并重写 onOptionsItemSelected()方法，对菜单项的选取状态进行处理，主要是用于根据菜单项的选取状态控制是否播放背景音乐。其具体代码如下：

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
```



```
        MenuInflater inflater=new MenuInflater(this);          //实例化一个 MenuInflater 对象
        inflater.inflate(R.menu.setting, menu);                //解析菜单文件
        return super.onCreateOptionsMenu(menu);
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getGroupId()==R.id.setting){                   //判断是否选择了参数设置菜单组
                if(item.isChecked()){                          //当菜单项已经被选中
                        item.setChecked(false);                //设置菜单项不被选中
                        Music.stop(this);
                }else{
                        item.setChecked(true);                 //设置菜单项被选中
                        Music.play(this, R.raw.jasmine);
                }
        }
        return true;
}
```



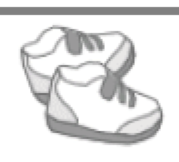
（7）编写 Music 类，在该类中首先声明一个 MediaPlayer 对象，然后编写用于播放背景音乐的 play()方法，最后再编写用于停止播放背景音乐的 stop()方法。其关键代码如下：

```
public class Music {
        private static MediaPlayer mp = null;                  //声明一个 MediaPlayer 对象
        public static void play(Context context, int resource) {
                stop(context);
                if (SettingsActivity.getBgSound(context)) {    //判断是否播放背景音乐
                        mp = MediaPlayer.create(context, resource);
                        mp.setLooping(true);                   //是否循环播放
                        mp.start();                            //开始播放
                }
        }
        public static void stop(Context context) {
                if (mp != null) {
                        mp.stop();                             //停止播放
                        mp.release();                          //释放资源
                        mp = null;
                }
        }
}
```

说明：

上面的代码中，加粗的代码 SettingsActivity.getBgSound(context)用于获取选项菜单存储的首选值，这样可以实现通过选项菜单控制是否播放背景音乐。

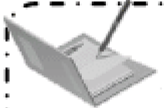
（8）编写 SettingsActivity 类，该类继承 PreferenceActivity 类，用于实现自动存储首选项的值。在 SettingsActivity 类中，首先重写 onCreate()方法，在该方法中调用 addPreferencesFromResource()方法加载首选项资源文件，然后编写获取是否播放背景音乐的首选项值的 getBgSound()



方法，在该方法中返回获取到的值。其关键代码如下：

```
public class SettingsActivity extends PreferenceActivity {
@Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.setting);
    }
    //获取是否播放背景音乐的首选项的值
    public static boolean getBgSound(Context context){
        return PreferenceManager.getDefaultSharedPreferences(context)
        .getBoolean("bgsound",true);
    }
}
```



说明：

PreferenceActivity 类用于实现对程序设置参数的存储。在该 Activity 中，设置参数的存储是完全自动的，不需要手动保存，非常方便。

（9）在 res 目录下创建一个 xml 目录，在该目录中添加一个名称为 setting.xml 的首选项资源文件。其具体代码如下：

```
<PreferenceScreen   xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="bgsound"
        android:title="播放背景音乐"
        android:summary="选中为播放背景音乐"
        android:defaultValue="true"/>
</PreferenceScreen>
```

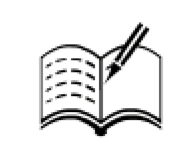
（10）在 MainActivity 中重写 onPause()方法，在该方法中调用 Music 类的 stop()方法停止播放背景音乐。其具体代码如下：

```
@Override
protected void onPause() {
    Music.stop(this);                         //停止播放背景音乐
    super.onPause();
}
```

（11）在 MainActivity 中重写 onResume()方法，在该方法中调用 Music 类的 play()方法开始播放背景音乐。其具体代码如下：

```
@Override
protected void onResume() {
    Music.play(this, R.raw.jasmine);          //播放背景音乐
    super.onResume();
}
```





16.3.2　制作开场动画

【例 16.6】 本实例主要在 Android 程序中实现制作开场动画的功能，运行程序，首先播放指定的视频，视频播放完毕后，进入到如图 16.6 所示的游戏主界面。

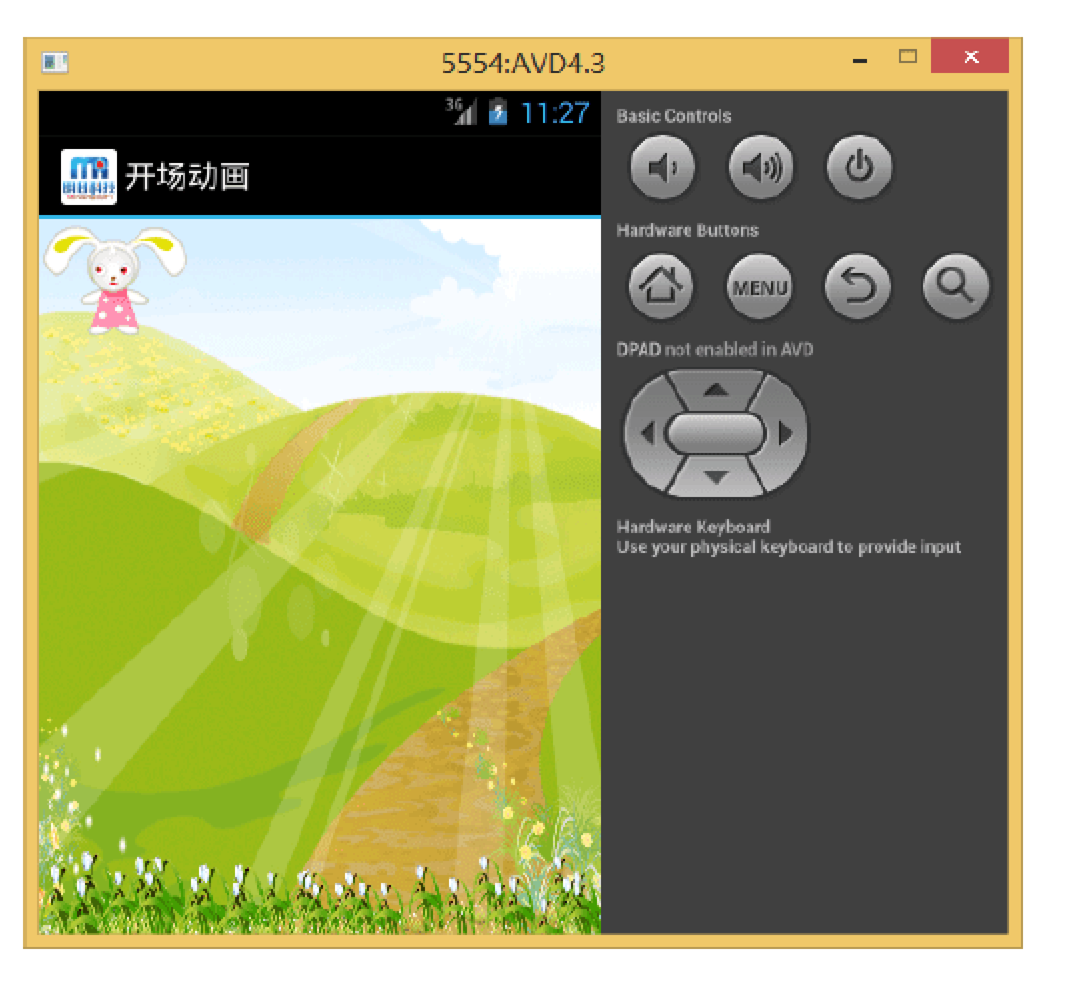


图 16.6　游戏主界面

实例位置：光盘\MR\Instance\16\16.6

程序的开发步骤如下：

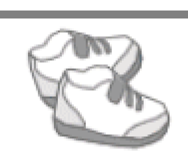
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的布局代码删除，然后添加一个 FrameLayout 帧布局管理器，并在该布局管理器中添加一个 ImageView，用于显示一只小兔子，另外，还需要为添加的帧布局管理器设置背景图片。

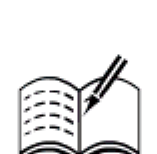
（2）在 res\layout 目录下创建一个布局文件 start.xml，在该文件中添加一居中显示的线性布局管理器，并在该布局管理器中添加一个 VideoView 组件，用于播放开场动画视频文件。其关键代码如下：

```
<VideoView
    android:id="@+id/video"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

（3）创建一个名称为 StartActivity 的 Activity，并重写其 onCreate()方法，在该方法中，首先获取 VideoView 组件，并获取要播放的文件对应的 URI，然后为 VideoView 组件指定要播放的视频，并让其获得焦点，再调用 start()方法开始播放视频，最后为 VideoView 添加完成事件监听器，在重写的 onCompletion()方法中调用 startMain()方法进入到游戏主界面。其具体代码如下：

```
video = (VideoView) findViewById(R.id.video);                    //获取 VideoView 组件
//获取要播放的文件对应的 URI
Uri uri = Uri.parse("android.resource://com.mingrisoft/"+R.raw.mingrisoft);
```






```
video.setVideoURI(uri);                                    //指定要播放的视频
video.requestFocus();                                      //让 VideoView 获得焦点
try {
    video.start();                                         //开始播放视频
    } catch (Exception e) {
        e.printStackTrace();                               //输出异常信息
    }
    //为 VideoView 添加完成事件监听器
    video.setOnCompletionListener(new OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
            startMain();                                   //进入游戏主界面
        }
    });
```

（4）编写进入游戏主界面的 startMain()方法，在该方法中创建一个新的 Intent 来启动游戏主界面的 Activity。其具体代码如下：

```
private void startMain(){                                              //进入游戏主界面
    Intent intent = new Intent(StartActivity.this, MainActivity.class); //创建 Intent
    startActivity(intent);                                             //启动新的 Activity
    StartActivity.this.finish();                                       //结束当前 Activity
}
```

（5）打开 AndroidManifest.xml 文件，在该文件中配置项目中应用的 Activity，这里首先将主 Activity 设置为 StartActivity，然后再配置 MainActivity。其关键代码如下：

```
<activity
    android:label="@string/app_name"
    android:name=".StartActivity" >
    <intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity android:name=".MainActivity"/>
```

16.4　本章常见错误

使用 MediaPlayer 和 SurfaceView 播放 MP4 视频时，出现视频无法播放的问题，该如何解决这个问题呢？

经过分析，发现 MediaPlayer 和 SurfaceView 组件只能播放 AVC（H264）编码格式的 MP4 视频文件，因此，要解决该问题，只需要将现有的 MP4 视频转换为 AVC（H264）编码格式（可以借助常用的视频格式转换软件转换）即可。





16.5　本 章 小 结

本章主要介绍了在 Android 中如何播放音频与视频等内容。需要重点说明的是两种播放音频的方法的区别，在本章中共介绍了两种播放音频的方法，一种是使用 MediaPlayer 播放，另一种是使用 SoundPool 播放，这两种方法的区别是：使用 MediaPlayer 每次只能播放一个音频，适用于播放长音乐或是背景音乐；使用 SoundPool 可以同时播放多个短小的音频，适用于播放按键音或者消息提示音等，希望读者根据实际情况选择合适的方法。

16.6　跟 我 上 机

参考答案：光盘\MR\跟我上机

在例 16.1 的基础上开发一个带音量控制功能的音乐播放器，运行程序，将显示一个带音量控制的音乐播放器，单击“播放”、“暂停/继续”和“停止”按钮，可以播放音乐、暂停/继续和停止音乐的播放，拖动“音量控制拖动条”上的滑块，可以调整音量的大小，并及时显示当前音量。音乐播放器界面布局如图 16.7 所示。

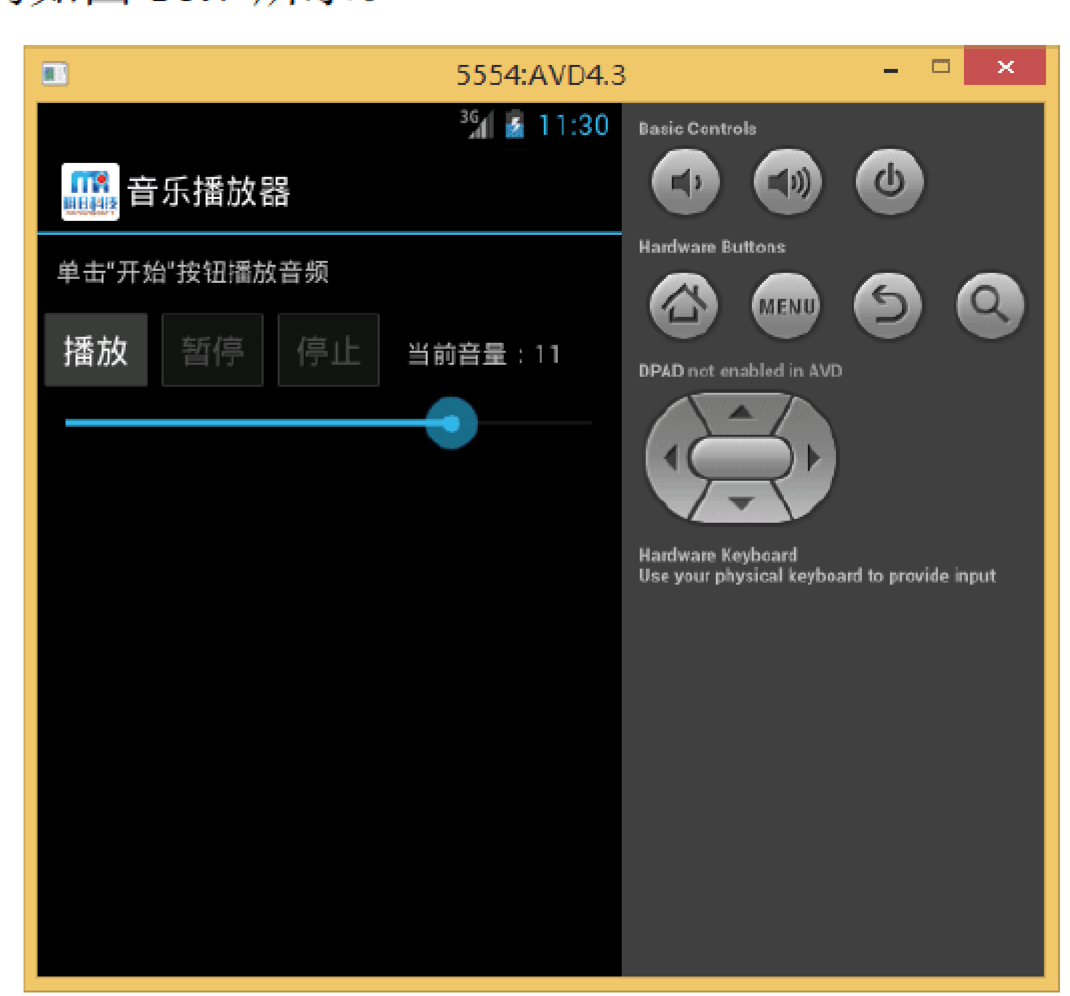
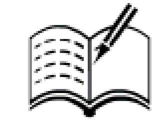


图 16.7　带音量控制的音乐播放器

具体实现时，首先需要在布局文件中添加一个拖动条组件，用来调整音量，然后在 onCreate()方法中，首先获取音频管理器类的对象，并设置当前调整音量只是针对媒体音乐进行；然后获取拖动条，并设置其最大值与当前值，同时获取显示当前音量的 TextView 组件，并设置其显示内容为当前音量；最后，为拖动条组件添加 OnSeekBarChangeListener 监听器，在重写的 onProgressChanged()方法中，显示改变后的音量，并将改变后的音量设置到音频管理器上，用来





改变音量的大小。调整音量大小的参考代码如下：

```
final AudioManager am = (AudioManager) MainActivity.this.getSystemService (Context.AUDIO_SERVICE);
                                                                              //获取音频管理类的对象
//设置当前调整音量只是针对媒体音乐
MainActivity.this.setVolumeControlStream(AudioManager.STREAM_MUSIC);
SeekBar seekbar = (SeekBar) findViewById(R.id.seekBar1);                      //获取拖动条
seekbar.setMax(am.getStreamMaxVolume(AudioManager.STREAM_MUSIC)); //设置拖动条的最大值
int progress=am.getStreamVolume(AudioManager.STREAM_MUSIC);      //获取当前的音量
seekbar.setProgress(progress);                                          //设置拖动条的默认值为当前音量
final TextView tv=(TextView)findViewById(R.id.volume);                  //获取显示当前音量的TextView组件
tv.setText("当前音量："+progress);                                        //显示当前音量
//为拖动条组件添加 OnSeekBarChangeListener 监听器
seekbar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}
    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress,boolean fromUser) {
        tv.setText("当前音量："+progress);                                  //显示改变后的音量
        am.setStreamVolume(AudioManager.STREAM_MUSIC,
                progress, AudioManager.FLAG_PLAY_SOUND);                //设置改变后的音量
    }
});
```

第 17 章

线程与消息处理

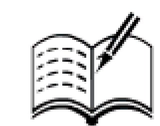
（视频讲解：35 分钟）

在程序开发时，对于一些比较耗时的操作，通常会为其开僻一个单独的线程来执行，这样可以尽可能减少用户的等待时间。在 Android 中，默认情况下，所有的操作都是在主线程中进行，这个主线程负责管理与 UI 相关的事件，而在自己创建的子线程中，又不能对 UI 组件进行操作，因此，Android 提供了消息处理传递机制来解决这一问题。本章将对 Android 中如何实现多线程以及如何通过线程和消息处理机制操作 UI 界面进行详细介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 通过实现 Runnable 接口创建、开启、休眠和中断线程
- ☐ 创建一个 Handler 对象发送并处理消息
- ☐ 开启新线程实现电子广告牌
- ☐ 多彩的霓虹灯
- ☐ 简易打地鼠游戏
- ☐ 开启一个新线程播放背景音乐





17.1 多线程的基本操作

在实际生活中，很多事情都是同时进行的，例如，可以一边看书，一边喝咖啡。而计算机则可以一边播放音乐，一边打印文档。对于这种可以同时进行的任务，可以用线程来表示，每个线程完成一个任务，并与其他线程同时执行，这种机制称为多线程。下面介绍如何创建线程、开启线程以及让线程休眠和中断线程。

17.1.1 创建线程

在 Android 中，提供了两种创建线程的方法，一种是通过 Thread 类的构造方法创建线程对象，并重写 run()方法实现；另一种是通过实现 Runnable 接口创建，下面分别进行介绍。

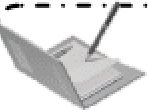
1. 通过 Thread 类的构造方法创建线程

在 Android 中，可以使用 Thread 类提供的以下构造方法来创建线程：

```
Thread(Runnable runnable)
```

该构造方法的参数 runnable，可以通过创建一个 Runnable 类的对象并重写其 run()方法来实现。例如，要创建一个名称为 thread 的线程，可以使用下面的代码：

```
Thread thread=new Thread(new Runnable(){
    //重写 run()方法
    @Override
    public void run() {
        //要执行的操作
    }
});
```



说明：

在 run()方法中，可以编写要执行的操作代码，当线程开启时，run()方法会被执行。

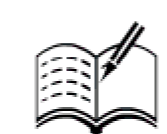
2. 通过实现 Runnable 接口创建线程

在 Android 中，还可以通过实现 Runnable 接口来创建线程。实现 Runnable 接口的语法格式如下：

```
public class ClassName extends Object implements Runnable
```

当一个类实现 Runnable 接口后，还需要实现其 run()方法。在 run()方法中可以编写要执行的操作的代码。





例如，要创建一个实现了 Runnable 接口的 Activity，可以使用下面的代码。

```
public class MainActivity extends Activity implements Runnable {
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
        @Override
        public void run() {
                //要执行的操作
        }
}
```

17.1.2 开启线程

创建线程对象后，还需要开启线程，线程才能执行。Thread 类提供了 start()方法，可以开启线程。其语法格式如下：

```
start()
```

例如，存在一个名称为 thread 的线程，如果想开启该线程，可以使用下面的代码。

```
thread.start();                          //开启线程
```

17.1.3 线程的休眠

线程的休眠就是让线程暂停多长时间后再次执行。同 Java 一样，在 Android 中也可以使用 Thread 类的 sleep()方法，让线程休眠指定的时间。sleep()方法的语法格式如下：

```
sleep(long time)
```

其中的参数 time 用于指定休眠的时间，单位为毫秒。

例如，想要线程休眠 1s，可以使用下面的代码：

```
Thread.sleep(1000);
```

17.1.4 中断线程

当需要中断指定线程时，可以使用 Thread 类提供的 interrupt()方法来实现。使用 interrupt()方法可以向指定的线程发送一个中断请求，并将该线程标记为中断状态。interrupt()方法的语法格式如下：

```
interrupt()
```

例如，存在一个名称为 thread 的线程，如果想中断该线程，可以使用下面的代码。

Note

```
…                                          //省略部分代码
thread.interrupt();
…                                          //省略部分代码
public void run() {
        while(!Thread.currentThread().isInterrupted()){
                …                          //省略部分代码
        }
}
```

另外，由于当线程执行 wait()、join()或者 sleep()方法时，线程的中断状态被清除，并且抛出 InterruptedException，所以，如果在线程中执行了 wait()、join()或者 sleep()方法，那么，想要中断线程时，就需要使用一个 boolean 型的标记变量来记录线程的中断状态，并通过该标记变量来控制循环的执行与停止。例如，通过名称为 isInterrupt 的 boolean 型变量来标记线程的中断。其关键代码如下：

```
private boolean isInterrupt=false;         //定义一个标记变量
        …                                  //省略部分代码
…                                          //在需要中断线程时，将 isInterrupt 的值设置为 true
public void run() {
        while(!isInterrupt){
                …                          //省略部分代码
        }
}
```

【例 17.1】 在 Eclipse 中创建 Android 项目，通过实现 Runnable 接口来创建线程、开启线程、让线程休眠指定时间和中断线程。

实例位置：光盘\MR\Instance\17\17.1

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加两个按钮，一个用于开启线程，另一个用于中断线程。

（2）打开默认添加的 MainActivity，让该类实现 Runnable 接口。修改后的创建类的代码如下：

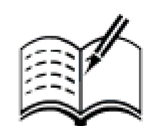
```
public class MainActivity extends Activity implements Runnable {}
```

（3）实现 Runnable 接口中的 run()方法，在该方法中，判断当前线程是否被中断，如果没有被中断，则将循环变量+1，并在日志中输出循环变量的值。其具体代码如下：

```
@Override
public void run() {
```



```
        while (!Thread.currentThread().isInterrupted()) {
            i++;
            Log.i("循环变量：", String.valueOf(i));
        }
}
```



Note

（4）在该 MainActivity 中，创建两个成员变量。其具体代码如下：

```
private Thread thread;                                  //声明线程对象
int i;                                                  //循环变量
```

（5）在 onCreate()方法中，首先获取布局管理器中添加的“开始”按钮，然后为该按钮添加单击事件监听器，在重写的 onCreate()方法中，根据当前 Activity 创建一个线程，并开启该线程。其具体代码如下：

```
Button startButton = (Button) findViewById(R.id.button1);  //获取“开始”按钮
startButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            i = 0;
            thread = new Thread(MainActivity.this);            //创建一个线程
            thread.start();                                    //开启线程
        }
});
```

（6）获取布局管理器中添加的“停止”按钮，并为其添加单击事件监听器，在重写的 onCreate()方法中，如果 thread 对象不为空，则中断线程，并向日志中输出提示信息。其具体代码如下：

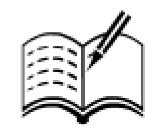
```
Button stopButton = (Button) findViewById(R.id.button2);   //获取“停止”按钮
stopButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if (thread != null) {
                thread.interrupt();                            //中断线程
                thread = null;
            }
            Log.i("提示：", "中断线程");
        }
});
```

（7）重写 MainActivity 的 onDestroy()方法，在该方法中中断线程。其具体代码如下：

```
@Override
protected void onDestroy() {
        if (thread != null) {
            thread.interrupt();                                //中断线程
            thread = null;
        }
        super.onDestroy();
}
```



运行本实例，在屏幕上将显示一个“开始”按钮和一个“停止”按钮，单击“开始”按钮，在日志面板中输出循环变量的值；单击“停止”按钮，将中断线程。日志面板的显示效果如图 17.1 所示。



Note

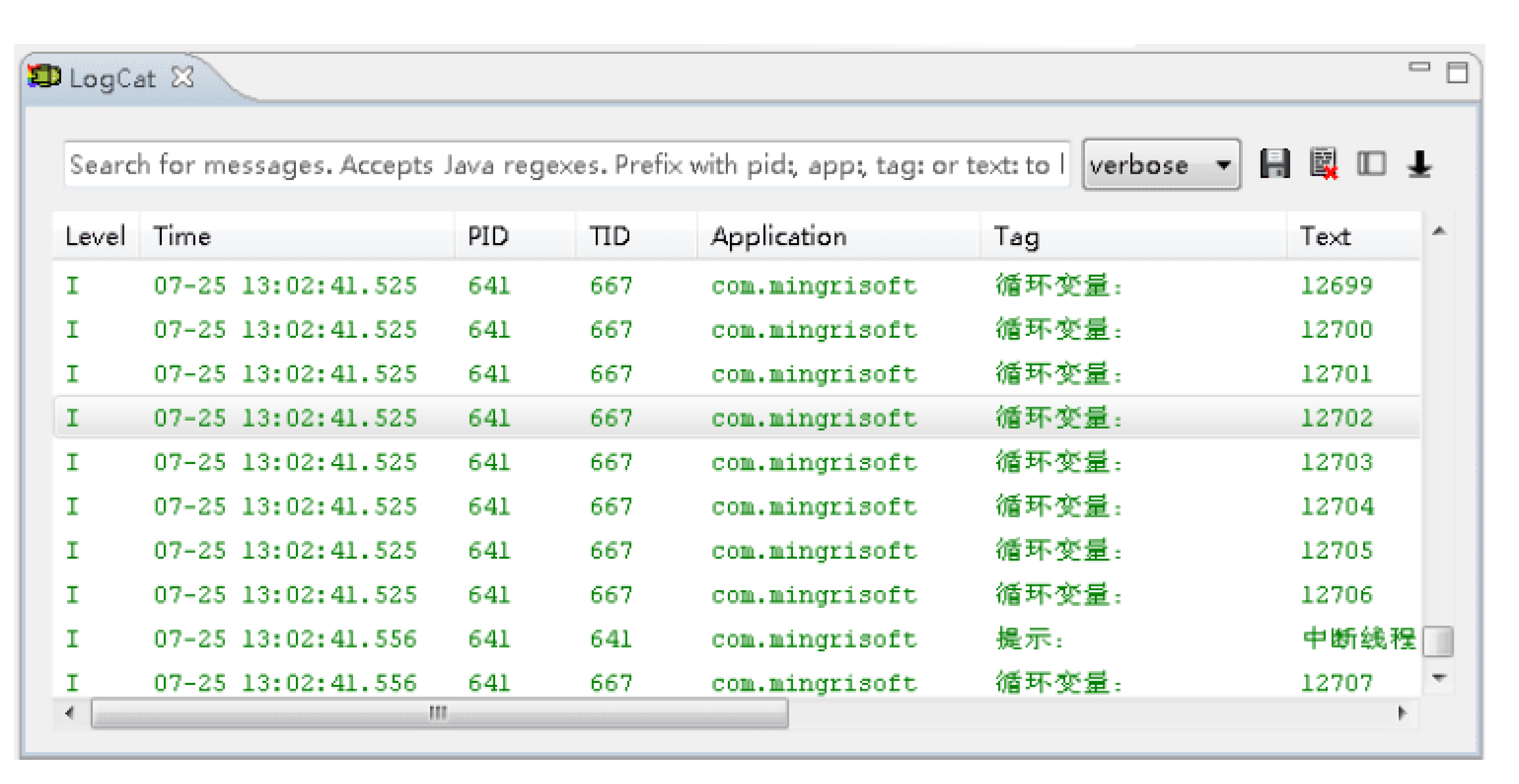


图 17.1　在日志面板中输出的内容

17.2　Handler 消息传递机制

17.1 节中已经介绍在 Android 中，如何创建、开启、休眠和中断线程。不过，还没有在新创建的子线程中对 UI 界面上的内容进行操作，如果应用前面介绍的方法对 UI 界面进行操作，将抛出异常。例如，在子线程的 run()方法中，循环修改文本框的显示文本，将抛出如图 17.2 所示的异常信息。

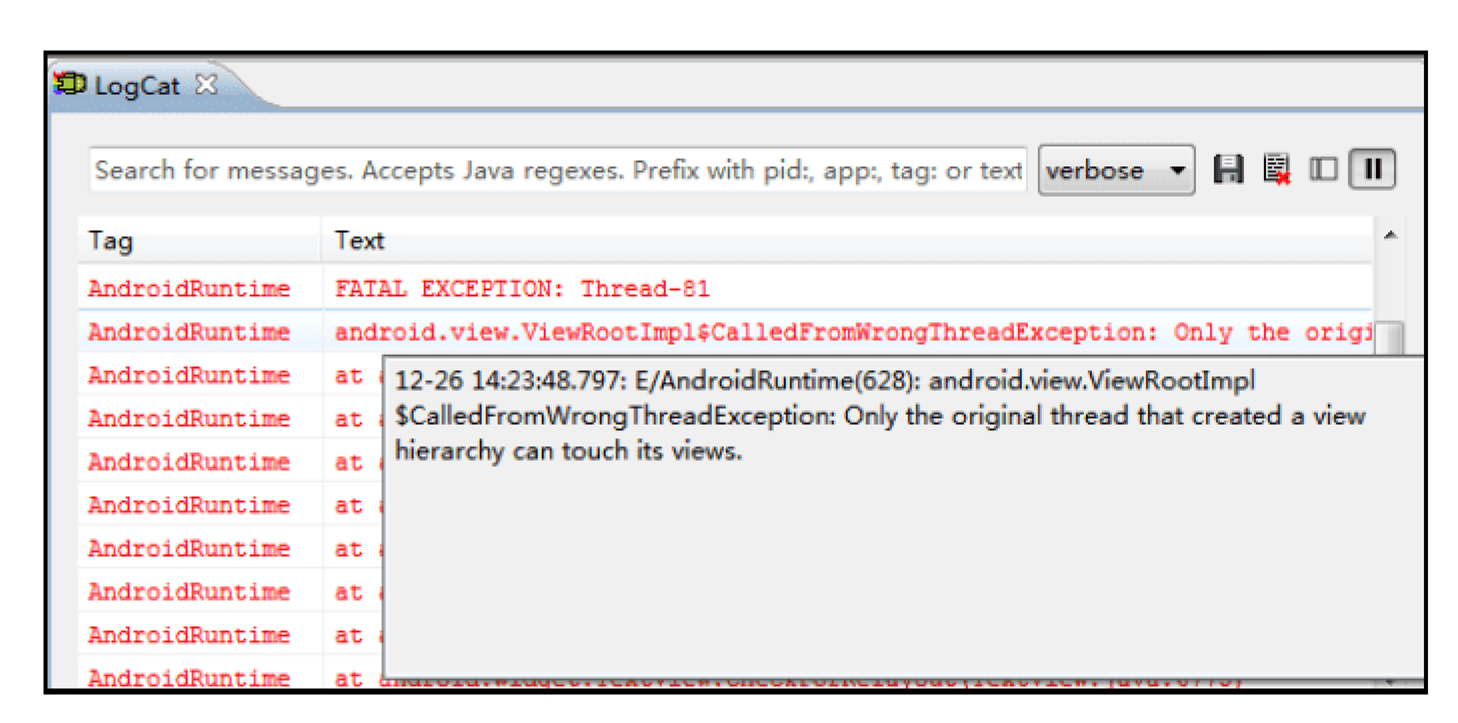


图 17.2　抛出的异常信息

为此，Android 中，引入了 Handler 消息传递机制，来实现在新创建的线程中操作 UI 界面。下面对 Handler 消息传递机制进行介绍。

17.2.1　循环者——Looper

在介绍 Lopper 之前，需要先来了解另一个概念，那就是 MessageQueue。在 Android 中，一



个线程对应一个 Looper 对象，而一个 Looper 对象又对应一个 MessageQueue（消息队列）。MessageQueue 用于存放 Message（消息），在 MessageQueue 中，存放的消息按照 FIFO（先进先出）原则执行，由于 MessageQueue 封装到 Looper 里面，所以这里不对 MessageQueue 进行过多介绍。

Looper 对象用来为一个线程开启一个消息循环，以操作 MessageQueue。默认情况下，Android 中新创建的线程是没有开启消息循环的，但是主线程除外，系统自动为主线程创建 Looper 对象，开启消息循环。所以，当在主线程中应用下面的代码创建 Handler 对象时，就不会出错，而如果在新创建的非主线程中，应用下面的代码创建 Handler 对象时，将产生如图 17.3 所示的异常信息。

```
Handler handler2 = new Handler();
```



图 17.3　在非主线程中创建 Handler 对象产生的异常信息

如果想要在非主线程中创建 Handler 对象，首先需要使用 Looper 类的 prepare()方法来初始化一个 Looper 对象，然后创建这个 Handler 对象，再使用 Looper 类的 loop()方法启动 Looper，从消息队列里获取和处理消息。

【例 17.2】 在 Eclipse 中创建 Android 项目，创建一个继承了 Thread 类的 LooperThread，并在重写的 run()方法中创建一个 Handler 对象发送并处理消息。

实例位置：光盘\MR\Instance\17\17.2

程序的开发步骤如下：

（1）创建一个继承了 Thread 类的 LooperThread，并在重写的 run()方法中创建一个 Handler 对象发送并处理消息。其关键代码如下：

```
public class LooperThread extends Thread {
    public Handler handler1;                          //声明一个 Handler 对象
    @Override
    public void run() {
        super.run();
        Looper.prepare();                             //初始化 Looper 对象
        //实例化一个 Handler 对象
        handler1 = new Handler() {
        public void handleMessage(Message msg) {
            Log.i("Looper",String.valueOf(msg.what));
            }
        };

        Message m=handler1.obtainMessage();           //获取一个消息
```

```
            m.what=0x11;                        //设置 Message 的 what 属性的值
            handler1.sendMessage(m);            //发送消息
            Looper.loop();                      //启动 Looper
        }
}
```

Note

（2）在 MainActivity 的 onCreate()方法中创建一个 LooperThread 线程，并开启该线程。其关键代码如下：

```
LooperThread thread=new LooperThread();        //创建一个线程
thread.start();                                //开启线程
```

运行本实例，在日志面板（LogCat）中输出如图 17.4 所示的内容。

```
I    07-25 13:13:41.275    1002    1177    com.mingrisoft    Looper    17
```

图 17.4　在日志面板（LogCat）中输出的内容

Looper 类提供的常用方法如表 17.1 所示。

表 17.1　Looper 类提供的常用方法

方　法	描　述
prepare()	用于初始化 Looper
loop()	调用 loop()方法后，Looper 线程就开始真正工作了，它会从消息队列里获取消息和处理消息
myLooper()	可以获取当前线程的 Looper 对象
getThread()	用于获取 Looper 对象所属的线程
quit()	用于结束 Looper 循环



注意：

写在 Looper.loop()之后的代码不会被执行，这个函数内部是一个循环，当调用 Handler.getLooper().quit()方法后，loop()方法才会中止，其后面的代码才能得以运行。

17.2.2　消息处理类——Handler

消息处理类（Handler）允许发送和处理 Message 或 Rannable 对象到其所在线程的 MessageQueue 中。Handler 有以下两个主要作用：

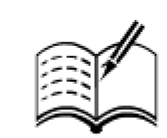
（1）将 Message 或 Runnable 应用 post()或 sendMessage()方法发送到 Message Queue 中，在发送时可以指定延迟时间、发送时间或者要携带的 Bundle 数据。当 MessageQueue 循环到该 Message 时，调用相应的 Handler 对象的 handlerMessage()方法对其进行处理。

（2）在子线程中与主线程进行通信，也就是在工作线程中与 UI 线程进行通信。

说明：

在一个线程中，只能有一个 Looper 和 MessageQueue，但是可以有多个 Handler，而且这些 Handler 可以共享同一个 Looper 和 MessageQueue。



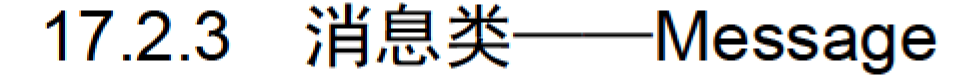


Handler 类提供的常用的发送和处理消息的方法如表 17.2 所示。

表 17.2　Handler 类提供的常用方法

方　法	描　述
handleMessage(Message msg)	处理消息的方法。通常重写该方法来处理消息，在发送消息时，该方法会自动回调
post(Runnable r)	立即发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
postAtTime(Runnable r, long uptimeMillis)	定时发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
postDelayed(Runnable r, long delayMillis)	延迟多少毫秒发送 Runnable 对象，该 Runnable 对象最后将被封装成 Message 对象
sendEmptyMessage(int what)	发送空消息
sendMessage(Message msg)	立即发送消息
sendMessageAtTime(Message msg, long uptimeMillis)	定时发送消息
sendMessageDelayed(Message msg, long delayMillis)	延迟多少毫秒发送消息

17.2.3　消息类——Message



消息类（Message）被存放在 MessageQueue 中，一个 MessageQueue 中可以包含多个 Message 对象。每个 Message 对象可以通过 Message.obtain()或者 Handler.obtainMessage()方法获得。一个 Message 对象具有如表 17.3 所示的 5 个属性。

表 17.3　Message 类的属性

属　性	类　型	描　述
arg1	int	用来存放整型数据
arg2	int	用来存放整型数据
obj	Object	用来存放发送给接收器的 Object 类型的任意对象
replyTo	Messenger	用来指定此 Message 发送到何处的可选 Messenger 对象
what	int	用于指定用户自定义的消息代码，这样接收者可以了解这个消息的信息

说明：

使用 Message 类的属性可以携带 int 型数据，如果要携带其他类型的数据，可以先将要携带的数据保存到 Bundle 对象中，然后通过 Message 类的 setDate()方法将其添加到 Message 中。

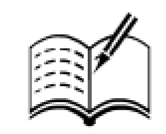
综上所述，Message 类的使用方法比较简单，只要在使用它时，注意以下 3 点即可：

☑ 尽管 Message 有 public 的默认构造方法，但是通常情况下，需要使用 Message.obtain()或 Handler.obtainMessage()方法来从消息池中获得空消息对象，以节省资源。

☑ 如果一个 Message 只需要携带简单的 int 型信息，应优先使用 Message.arg1 和 Message.arg2 属性来传递信息，这比用 Bundle 更省内存。



☑ 尽可能使用 Message.what 来标识信息，以便用不同方式处理 Message。



Note

17.3 综 合 应 用

17.3.1 开启新线程实现电子广告牌

【例 17.3】 在 Eclipse 中创建一个 Android 项目，开启新线程实现电子广告牌的功能，运行本实例，在屏幕上将每隔 2s 随机显示一张广告，如图 17.5 所示。



图 17.5 电子广告牌

实例位置：光盘\MR\Instance\17\17.3

程序的开发步骤如下：

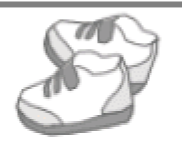
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认添加的 TextView 组件上方添加一个 ImageView 组件，用于显示广告图片，并设置垂直线程布局管理器内的组件水平居中显示。

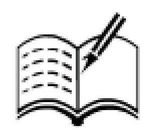
（2）打开默认添加的 MainActivity，让该类实现 Runnable 接口。修改后的创建类的代码如下：

```
public class MainActivity extends Activity implements Runnable {}
```

（3）实现 Runnable 接口中的 run()方法，在该方法中判断当前线程是否被中断，如果没有被中断，则首先产生一个随机数，然后获取一个 Message，并将要显示的广告图片的索引值和对应标题保存到该 Message 中，再发送消息，最后让线程休眠 2 秒钟。其具体代码如下：

```
@Override
public void run() {
```





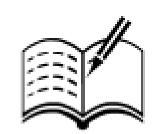

```
        int index = 0;
        while (!Thread.currentThread().isInterrupted()) {
            index = new Random().nextInt(path.length);          //产生一个随机数
            Message m = handler.obtainMessage();                //获取一个 Message
            m.arg1 = index;                                     //保存要显示广告图片的索引值
            Bundle bundle = new Bundle();                       //获取 Bundle 对象
            m.what = 0x101; //设置消息标识
            bundle.putString("title", title[index]);            //保存标题
            m.setData(bundle);                                  //将 Bundle 对象保存到 Message 中
            handler.sendMessage(m);                             //发送消息
            try {
                Thread.sleep(2000);                             //线程休眠 2 秒钟
            } catch (InterruptedException e) {
                e.printStackTrace();                            //输出异常信息
            }
        }
}
```

（4）在该 MainActivity 中，创建程序中所需的成员变量。其具体代码如下：

```
private ImageView iv;                                           //声明一个显示广告图片的 ImageView 对象
private Handler handler;                                        //声明一个 Handler 对象
private int[] path = new int[] { R.drawable.img01, R.drawable.img02,
        R.drawable.img03, R.drawable.img04, R.drawable.img05,
        R.drawable.img06 };                                     //保存广告图片的数组
private String[] title = new String[] { "编程词典系列产品", "高效开发", "快乐分享", "用户人群","快速学习", "
全方位查询" };                                                   //保存显示标题的数组
```

（5）在 onCreate()方法中，首先获取布局管理器中添加的 ImageView 组件，然后创建一个新线程，并开启该线程，最后再实例化一个 Handler 对象，在重写的 handleMessage()方法中，更新 UI 界面中的 ImageView 和 TextView 组件。其具体代码如下：

```
iv = (ImageView) findViewById(R.id.imageView1);                 //获取显示广告图片的 ImageView
Thread t = new Thread(this);                                    //创建新线程
t.start();                                                      //开启线程
handler = new Handler() {                                       //实例化一个 Handler 对象
    @Override
    public void handleMessage(Message msg) {
        //更新 UI
        TextView tv = (TextView) findViewById(R.id.textView1);    //获取 TextView 组件
        if (msg.what == 0x101) {
            tv.setText(msg.getData().getString("title"));//设置标题
            iv.setImageResource(path[msg.arg1]);        //设置要显示的图片
        }
        super.handleMessage(msg);
    }
};
```


Note

17.3.2 多彩的霓虹灯

【例 17.4】 本实例要求在 Android 程序中实现多彩霓虹灯的效果，运行程序，将在 Android 窗口中显示一个多彩的霓虹灯，它可以不断地变换颜色，如图 17.6 所示。

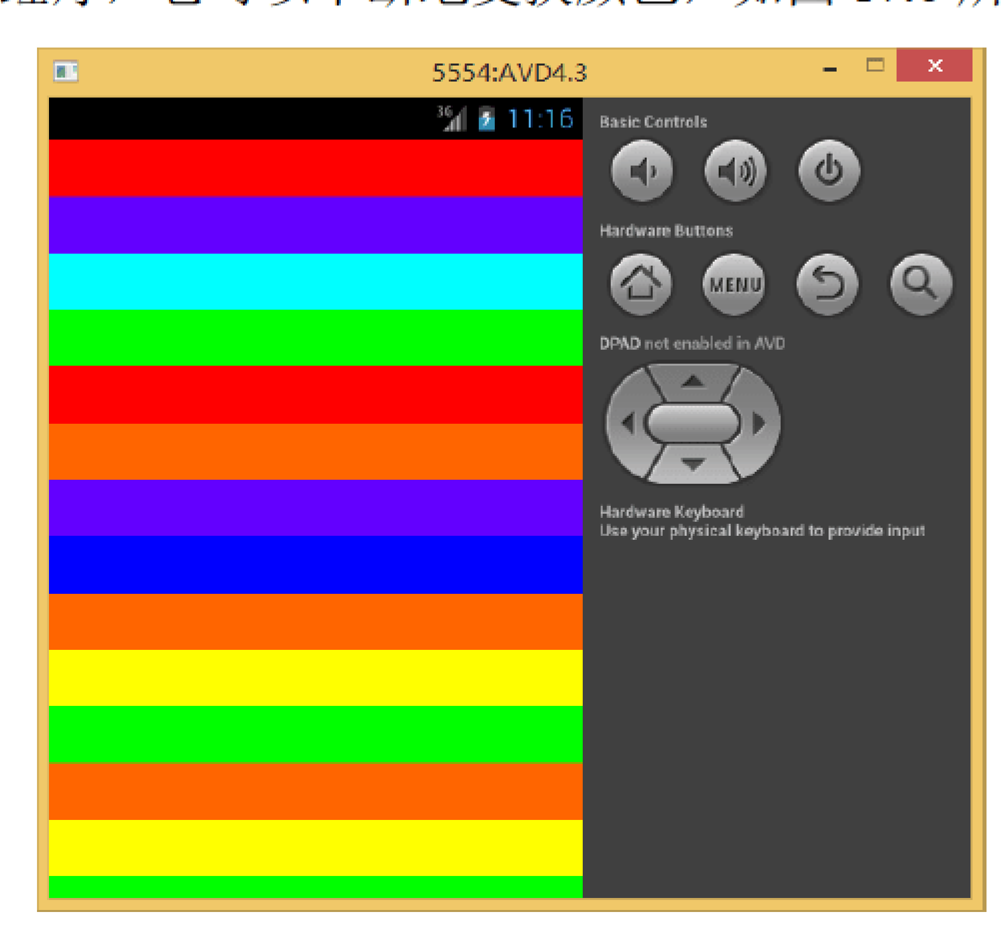


图 17.6　多彩的霓虹灯

实例位置：光盘\MR\Instance\17\17.4

程序的开发步骤如下：

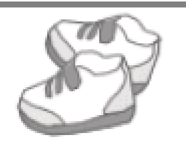
（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，并为默认添加的线性布局管理器设置 ID 属性。

（2）在 res\values\目录下，创建一个保存颜色资源的 colors.xml 文件。在该文件中，定义 7 个颜色资源，名称依次为 color1，color2，…，color7，分别代表赤、橙、黄、绿、青、蓝、紫所对应的颜色值。colors.xml 文件的关键代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="color1">#ffff0000</color>
    <color name="color2">#ffff6600</color>
    <color name="color3">#ffffff00</color>
    <color name="color4">#ff00ff00</color>
    <color name="color5">#ff00ffff</color>
    <color name="color6">#ff0000ff</color>
    <color name="color7">#ff6600ff</color>
</resources>
```

（3）在该 MainActivity 中，声明程序中所需的成员变量。其具体代码如下：

```
private Handler handler;                          //创建 Handler 对象
private static LinearLayout linearLayout;         //整体布局
public static TextView[] tv = new TextView[14];   //TextView 数组
```



```
int[] bgColor=new int[]{R.color.color1,R.color.color2,R.color.color3,
            R.color.color4,R.color.color5,R.color.color6,R.color.color7};     //使用颜色资源
private int index=0;                                                          //当前颜色值
```

（4）在 MainActivity 的 onCreate()方法中，首先获取线程布局管理器，然后获取屏幕的高度，接下来再通过一个 for 循环创建 14 个文本框组件，并添加到线性布局管理器中。其具体代码如下：

```
linearLayout=(LinearLayout)findViewById(R.id.ll);                     //获取线性布局管理器
int height=this.getResources().getDisplayMetrics().heightPixels;      //获取屏幕的高度
for(int i=0;i<tv.length;i++){
      tv[i]=new TextView(this);                                       //创建一个文本框对象
      tv[i].setWidth(this.getResources().getDisplayMetrics().widthPixels);  //设置文本框宽度
      tv[i].setHeight(height/tv.length);                              //设置文本框高度
      linearLayout.addView(tv[i]);                      //将 TextView 组件添加到线性布局管理器中
}
```

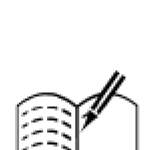
（5）创建并开启一个新线程，在重写的 run()方法中实现一个循环。在该循环中，首先获取一个 Message 对象，并为其设置一个消息标识，然后发送消息，最后让线程休眠 1 秒钟。其具体代码如下：

```
Thread t = new Thread(new Runnable(){
      @Override
      public void run() {
            while (!Thread.currentThread().isInterrupted()) {
                  Message m = handler.obtainMessage();                //获取一个 Message
                  m.what=0x101;                                       //设置消息标识
                  handler.sendMessage(m);                             //发送消息
                  try {
                        Thread.sleep(new Random().nextInt(1000));     //休眠 1 秒钟
                  } catch (InterruptedException e) {
                        e.printStackTrace();                          //输出异常信息
                  }
            }
      }
});
t.start();                                                            //开启线程
```

（6）创建一个 Handler 对象，在重写的 handleMessage()方法中，为每个文本框设置背景颜色，该背景颜色从颜色数组中随机获取。其具体代码如下：

```
handler = new Handler() {
      @Override
      public void handleMessage(Message msg) {
            int temp=0;                                               //临时变量
            if (msg.what == 0x101) {
                  for(int i=0;i<tv.length;i++){
                        temp=new Random().nextInt(bgColor.length);    //产生一个随机数
                        //去掉重复的并且相邻的颜色
                        if(index==temp){
```





Note

```
                temp++;
                if(temp==bgColor.length){
                    temp=0;
                }
            }
            index=temp;
            //为文本框设置背景
            tv[i].setBackgroundColor(getResources().getColor(bgColor[index]));
        }
    }
    super.handleMessage(msg);
  }
};
```

（7）在 AndroidManifest.xml 文件的<activity>标记中，设置 android:theme 属性，实现全屏显示。其关键代码如下：

```
android:theme="@android:style/Theme.Black.NoTitleBar"
```

17.3.3　简易打地鼠游戏

【例 17.5】 本实例要求使用多线程技术开发一个简单的打地鼠游戏，运行程序，在屏幕上将随机显示地鼠，触摸地鼠后，该地鼠将不显示，同时在屏幕上通过消息提示框显示打到了几只地鼠，如图 17.7 所示。

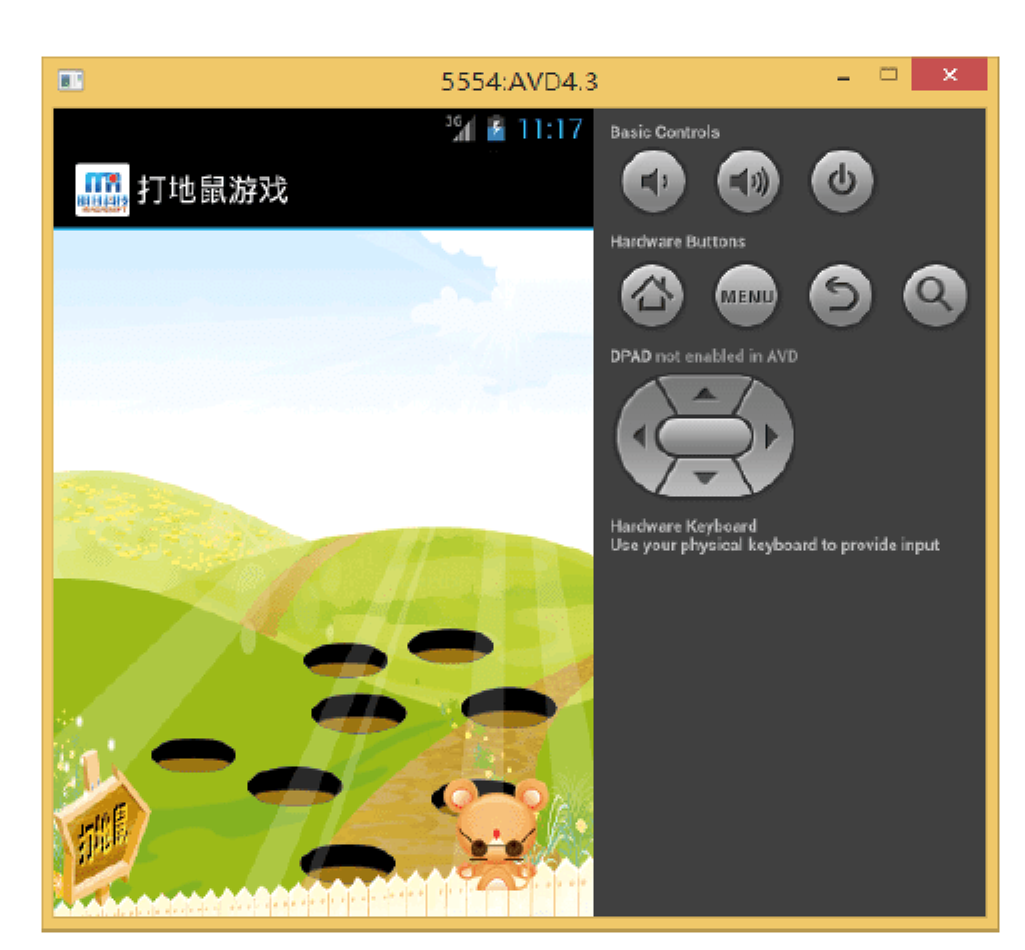


图 17.7　简易打地鼠游戏

实例位置：光盘\MR\Instance\17\17.5



说明：

运行本实例时，最好将 Android 模拟器（AVD）修改为 WSVGA 模式。



程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，首先将默认添加的布局管理器和 TextView 组件删除，然后添加一个帧布局管理器，最后在该布局管理器中添加一个用于显示地鼠的 ImageView 组件，并设置其显示一张地鼠图片。其关键代码如下：

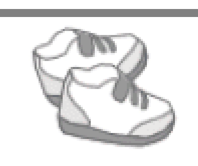
```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fl"
    android:background="@drawable/background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/mouse" />
</FrameLayout>
```

（2）在 MainActivity 中，声明程序中所需的成员变量。其代码如下：

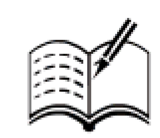
```
private int i = 0;                                  //记录其打到了几只地鼠
private ImageView mouse;                            //声明一个 ImageView 对象
private Handler handler;                            //声明一个 Handler 对象
public int[][] position = new int[][] { { 231, 325 }, { 424, 349 },
    { 521, 256 }, { 543, 296 }, { 719, 245 }, { 832, 292 },
    { 772, 358 } };                                 //创建一个表示地鼠位置的数组
```

（3）创建并开启一个新线程，在重写的 run()方法中，创建一个记录地鼠位置的索引值的变量，并实现一个循环。在该循环中，首先生成一个随机数，并获取一个 Message 对象，然后将生成的随机数作为地鼠位置的索引值保存到 Message 对象中，再为该 Message 设置一个消息标识，并发送消息，最后让线程休眠一段时间（该时间随机产生）。其代码如下：

```
Thread t = new Thread(new Runnable() {
    @Override
    public void run() {
        int index = 0;                                      //创建一个记录地鼠位置的索引值
        while (!Thread.currentThread().isInterrupted()) {
            index = new Random().nextInt(position.length);  //产生一个随机数
            Message m = handler.obtainMessage();            //获取一个 Message
            m.arg1 = index;                                 //保存地鼠位置的索引值
            m.what = 0x101;                                 //设置消息标识
            handler.sendMessage(m);                         //发送消息
            try {
                Thread.sleep(new Random().nextInt(500) + 500);    //休眠一段时间
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
```



```
});
t.start();                                                    //开启线程
```



Note

（4）创建一个 Handler 对象，在重写的 handleMessage()方法中，首先定义一个记录地鼠位置索引值的变量，然后使用 if 语句根据消息标识判断是否为指定的消息，如果是，则获取消息中保存的地鼠位置的索引值，并设置地鼠在指定位置显示。其具体代码如下：

```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        int index = 0;
        if (msg.what == 0x101) {
            index = msg.arg1;                                 //获取位置索引值
            mouse.setX(position[index][0]);                   //设置 X 轴位置
            mouse.setY(position[index][1]);                   //设置 Y 轴位置
            mouse.setVisibility(View.VISIBLE);                //设置地鼠显示
        }
        super.handleMessage(msg);
    }
};
```

（5）获取布局管理器中添加的 ImageView 组件，并为该组件添加触摸监听器，在重写的 onTouch()方法中，首先设置地鼠不显示，然后将 i 的值+1，再通过消息提示框显示打到了几只地鼠。其代码如下：

```
mouse = (ImageView) findViewById(R.id.imageView1);            //获取 ImageView 对象
mouse.setOnTouchListener(new OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        v.setVisibility(View.INVISIBLE);                      //设置地鼠不显示
        i++;
        Toast.makeText(MainActivity.this, "打到[ " + i + " ]只地鼠！ ",
                Toast.LENGTH_SHORT).show();                   //显示消息提示框
        return false;
    }
});
```

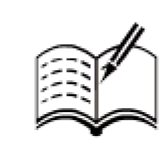
17.4　本章常见错误

在 Android 程序中，UI 线程是主线程，理论上来讲，Activity 中调用了 finish()方法后，主线程就终止了，那么其子线程也应该停止，但是通过具体的实例测试，发现调用 finish()方法退出程序后，子线程还在后台运行，那么，如何解决这个问题呢？

有两种方法可以避免上面所描述的问题。

☑　退出程序时，将线程内循环的标识符修改为 false。





☑　不采用 while(isRun){}这样的循环方式，而是通过 Android 中的 Handler 机制实现。例如：

```
handler.post(test);
Runnable test=new Runnable(){
      public void run(){
      handler.postDelayed(test,1000);
      }
}
//退出
handler.removeCallbacks(test);
```

17.5　本章小结

本章主要介绍了在 Android 中如何实现多线程。由于在 Android 中，不能在子线程（也称为工作线程）中更新主线程（也称为 UI 线程）中的 UI 组件。因此，Android 引入了消息传递机制，通过使用 Looper、Handler 和 Message 就可以轻松实现多线程中更新 UI 界面的功能了，这与 Java 中的多线程不同，希望读者能很好地理解，做到灵活应用。另外，多线程在游戏开发时，是非常重要的一项技术。

17.6　跟我上机

参考答案：光盘\MR\跟我上机

开发一个 Android 程序实现的功能为：开启一个新线程播放背景音乐，在音乐文件播放完毕后，暂停 5s 后重新开始播放。在程序布局文件中，将默认添加的 TextView 组件删除，然后添加一个“开始”按钮，用于开启线程并播放背景音乐。具体实现时，需要编写一个自定义的 playBGSound()方法，该方法中，首先判断 MediaPlayer 对象是否为空，如果不为空，则释放该对象，然后创建一个用于播放背景音乐的 MediaPlayer 对象，并开始播放，最后为该 MediaPlayer 对象添加播放完成事件监听器，在重写的 onCompletion()方法中，让线程休眠 5s，并调用 playBGSound()方法重新播放音乐。其参考代码如下：

```
private void playBGSound() {
      if (mp != null) {
            mp.release();                                              //释放资源
      }
      mp = MediaPlayer.create(MainActivity.this, R.raw.jasmine);
      mp.start();                                                      //开始播放
      mp.setOnCompletionListener(new OnCompletionListener(){      //为 MediaPlayer 添加播放完成事件
            @Override
            public void onCompletion(MediaPlayer mp) {
```

```
                try {
                    Thread.sleep(5000);                          //线程休眠 5 秒钟
                    playBGSound();                               //重新播放音乐
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
}
```

在 onCreate()方法中，获取布局管理器中添加的“开始”按钮，并为该按钮添加单击事件监听器，在重写的 onClick()方法中，首先设置该按钮不可用，然后创建一个用于播放背景音乐的线程，并开启该线程，在重写的 run()方法中，调用 playBGSound()方法播放背景音乐。其参考代码如下：

```
Button button = (Button) findViewById(R.id.button1);         //获取布局管理器中添加的“开始”按钮
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        ((Button) v).setEnabled(false);                       //设置按钮不可用
        thread = new Thread(new Runnable() {                 //创建一个用于播放背景音乐的线程
            @Override
            public void run() {
                playBGSound();                               //播放背景音乐
            }
        });
        thread.start();                                      //开启线程
    }
});
```



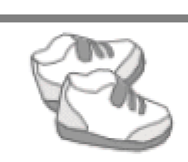
网络编程技术

（视频讲解：1 小时 6 分钟）

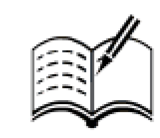
Google 公司是以网络搜索引擎白手起家的，通过大胆的创意和不断的研发努力，目前已经成为网络世界的巨头，而出自于 Google 之手的 Android 平台，在进行网络编程和 Internet 应用上，也是非常优秀的。本章将对 Android 中的网络编程及 Internet 应用的相关知识进行详细介绍。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 向服务器发送 GET 请求
- ☐ 向服务器发送 POST 请求
- ☐ 使用 HttpClient 向服务器发送 GET 请求
- ☐ 应用 HttpClient 向服务器发送 POST 请求
- ☐ 使用 WebView 组件浏览网页
- ☐ 使用 WebView 组件加载 HTML 代码
- ☐ 让 WebView 允许执行 JavaScript
- ☐ 打造功能实用的网页浏览器
- ☐ 获取天气预报



18.1 通过 HTTP 访问网络


Note

随着智能手机和平板电脑等移动终端设备的迅速发展，现在的 Internet 已经不再只是传统的有线互联网，它还包括了移动互联网。同有线互联网一样，移动互联网也可以使用 HTTP 访问网络。在 Android 中，针对 HTTP 进行网络通信的方法主要有两种：一种是使用 HttpURLConnection 实现，另一种是使用 HttpClient 实现。下面分别进行介绍。

18.1.1 使用 HttpURLConnection 访问网络

HttpURLConnection 类位于 java.net 包中，用于发送 HTTP 请求和获取 HTTP 响应。由于该类是抽象类，不能直接实例化对象，需要使用 URL 的 openConnection()方法来获得。例如，要创建一个 http://www.mingribook.com 网站对应的 HttpURLConnection 对象，可以使用下面的代码。

```
URL url = new URL("http://www.mingribook.com/");
HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
```



说明：

通过 openConnection()方法创建的 HttpURLConnection 对象，并没有真正地执行连接操作，只是创建了一个新的实例，在进行连接前，还可以设置一些属性。例如，连接超时的时间和请求方式等。

创建了 HttpURLConnection 对象后，就可以使用该对象发送 HTTP 请求了。HTTP 请求通常分为 GET 请求和 POST 请求两种，下面分别进行介绍。

1. 发送 GET 请求

使用 HttpURLConnection 对象发送请求时，默认发送的就是 GET 请求。因此，发送 GET 请求比较简单，只需要在指定连接地址时，先将要传递的参数通过“?参数名=参数值”进行传递（多个参数间使用英文半角的逗号分隔。例如，要传递用户名和 E-mail 地址两个参数可以使用“?user=wgh,email=wgh717@sohu.com”实现），然后获取流中的数据，并关闭连接即可。

下面通过一个具体的实例来说明如何使用 HttpURLConnection 发送 GET 请求。

【例 18.1】 在 Eclipse 中创建 Android 项目，实现向服务器发送 GET 请求，并获取服务器的响应结果。

实例位置：光盘\MR\Instance\18\18.1

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加一个 id 为 content 的编辑框（用于输入微博内容）



和一个“发表”按钮，再添加一个滚动视图，并在该视图中添加一个线性布局管理器，最后还需要在该线性布局管理器中添加一个文本框，用于显示从服务器上读取的微博内容。其关键代码如下：

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical" >
    <EditText
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button" />
    <ScrollView
        android:id="@+id/scrollView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1" >
        <LinearLayout
            android:id="@+id/linearLayout1"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
            <TextView
                android:id="@+id/result"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_weight="1" />
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

（2）在该 MainActivity 中，创建程序中所需的成员变量。其具体代码如下：

```
private EditText content;                //声明一个输入文本内容的编辑框对象
private Button button;                   //声明一个发表按钮对象
private Handler handler;                 //声明一个 Handler 对象
private String result = "";              //声明一个代表显示内容的字符串
private TextView resultTV;               //声明一个显示结果的文本框对象
```

（3）编写一个无返回值的 send()方法，用于建立一个 HTTP 连接，并将输入的内容发送到 Web 服务器上，再读取服务器的处理结果。其具体代码如下：

```
public void send() {
    String target="";
```

Note

```
        target = "http://192.168.1.66:8081/blog/index.jsp?content="
                        +base64(content.getText().toString().trim());                    //要访问的 URL 地址
        URL url;
        try {
            url = new URL(target);                                                       //创建 URL 对象
            HttpURLConnection urlConn = (HttpURLConnection) url
                        .openConnection();                                               //创建一个 HTTP 连接
            InputStreamReader in = new InputStreamReader(
                        urlConn.getInputStream());                                       //获得读取的内容
            BufferedReader buffer = new BufferedReader(in);                              //获取输入流对象
            String inputLine = null;
            //通过循环逐行读取输入流中的内容
            while ((inputLine = buffer.readLine()) != null) {
                result += inputLine + "\n";
            }
            in.close();                                                                  //关闭字符输入流对象
            urlConn.disconnect();                                                        //断开连接
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
}
```

（4）在应用 GET 方法传递中文的参数时会产生乱码，这时可以采用对其进行 Base64 编码来解决，为此，需要编写一个 base64()方法，对要进行传递的参数进行 Base64 编码。base64()方法的具体代码如下：

```
public String base64(String content){
        try {
            //对字符串进行 Base64 编码
            content=Base64.encodeToString(content.getBytes("utf-8"), Base64.DEFAULT);
            content=URLEncoder.encode(content);                                          //对字符串进行 URL 编码
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();                                                         //输出异常信息
        }
        return content;
}
```

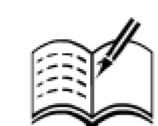


说明：

要解决应用 GET 方法传递中文参数乱码的情况，也可以不采用 Base64 编码来解决，而使用 Java 提供的 URLEncoder 类来实现。

（5）在 onCreate()方法中，获取布局管理器中添加用于输入内容的编辑框，用于显示结果的文本框和“发表”按钮，并为“发表”按钮添加单击事件监听器。在重写的 onClick()方法中，首先判断输入的内容是否为空，如果为空则给出消息提示，否则，创建一个新的线程，调用 send()





方法发送并读取微博信息。其具体代码如下：

```
content = (EditText) findViewById(R.id.content);          //获取输入文本内容的 EditText 组件
resultTV = (TextView) findViewById(R.id.result);          //获取显示结果的 TextView 组件
button = (Button) findViewById(R.id.button);              //获取“发表”按钮组件
//为按钮添加单击事件监听器
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        if ("".equals(content.getText().toString())) {
            Toast.makeText(MainActivity.this, "请输入要发表的内容！",
                    Toast.LENGTH_SHORT).show();           //显示消息提示
            return;
        }
        //创建一个新线程，用于发送并读取微博信息
        new Thread(new Runnable() {
            public void run() {
                send();                                   //发送文本内容到 Web 服务器，并读取
                Message m = handler.obtainMessage();      //获取一个 Message
                handler.sendMessage(m);                   //发送消息
            }
        }).start();                                       //开启线程
    }
});
```

（6）创建一个 Handler 对象，在重写的 handleMessage()方法中，当变量 result 不为空时，将其显示到结果文本框中，并清空编辑器。其具体代码如下：

```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (result != null) {
            resultTV.setText(result);                     //显示获得的结果
            content.setText("");                          //清空编辑框
        }
        super.handleMessage(msg);
    }
};
```

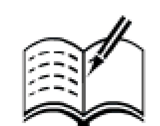
（7）由于在本实例中，需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

另外，还需要编写一个 Java Web 实例，用于接收 Android 客户端发送的请求，并做出响应，这里编写一个名称为 index.jsp 的文件。在该文件中，首先获取参数 content 指定的微博信息，并保存到变量 content 中，然后替换变量 content 中的加号，这是由于在进行 URL 编码时，将加号



转换为%2B 了，最后再对 content 进行 Base64 解码，并输出转码后的 content 变量的值。其具体代码如下：

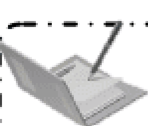


```
<%@ page contentType="text/html; charset=utf-8" language="java" import="sun.misc.BASE64Decoder"%>
<%
 String content="";
 if(request.getParameter("content")!=null){
      content=request.getParameter("content");                  //获取输入的微博信息
//替换 content 中的加号，这是由于在进行 URL 编码时，将+号转换为%2B 了
      content=content.replaceAll("%2B","+");
      BASE64Decoder decoder=new BASE64Decoder();
      content=new String(decoder.decodeBuffer(content),"utf-8");     //进行 base64 解码
 }
%>
<%="发表一条微博，内容如下："%>
<%=content%>
```

将 index.jsp 文件放到 Tomcat 安装路径下的 webapps\blog 目录下，并启动 Tomcat 服务器。然后运行本实例，在屏幕上方的编辑框中输入一条微博信息，并单击“发表”按钮，在下方将显示 Web 服务器的处理结果。例如，输入“坚持到底就是胜利！”后，单击“发表”按钮，将显示如图 18.1 所示的运行效果。



图 18.1　使用 GET 方式发表并显示微博信息



说明：

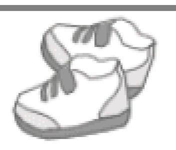
运行本章的所有程序时，都需要将 Android 模拟器（AVD）修改为 WSVGA 模式。

2. 发送 POST 请求

由于采用 GET 方式发送请求只适合发送大小在 1024B 以内的数据，所以当要发送的数据比较大时，就需要使用 POST 方式。在 Android 中，使用 HttpURLConnection 类在发送请求时，默认采用的是 GET 请求，如果要发送 POST 请求，需要通过其 setRequestMethod()方法进行指定。例如，创建一个 HTTP 连接，并为该连接指定请求的发送方式为 POST，可以使用下面的代码。

```
HttpURLConnection urlConn = (HttpURLConnection) url.openConnection();     //创建一个 HTTP 连接
urlConn.setRequestMethod("POST");                                          //指定请求方式为 POST
```

发送 POST 请求要比发送 GET 请求复杂一些，它经常需要通过 HttpURLConnection 类及其父类 URLConnection 提供的如表 18.1 所示的方法设置相关内容。



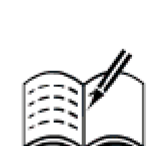


表 18.1　发送 POST 请求时常用的方法

方　　法	描　　述
setDoInput(boolean newValue)	用于设置是否向连接中写入数据，如果参数值为 true 时，表示写入数据，否则不写入数据
setDoOutput(boolean newValue)	用于设置是否从连接中读取数据，如果参数值为 true 时，表示读取数据，否则不读取数据
setUseCaches(boolean newValue)	用于设置是否缓存数据，如果参数值为 true，表示缓存数据，否则表示禁用缓存
setInstanceFollowRedirects(boolean followRedirects)	用于设置是否应该自动执行 HTTP 重定向，参数值为 true 时，表示自动执行，否则不自动执行
setRequestProperty(String field, String newValue)	用于设置一般请求属性，例如，要设置内容类型为表单数据，可以进行以下设置 setRequestProperty("Content-Type","application/x-www-form-urlencoded")

下面将通过一个具体的实例来介绍如何使用 HttpURLConnection 类发送 POST 请求。

【例 18.2】 在 Eclipse 中创建 Android 项目，实现向服务器发送 POST 请求，并获取服务器的响应结果。

实例位置：光盘\MR\Instance\18\18.2

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加一个 id 为 content 的编辑框（用于输入微博内容）和一个“发表”按钮，最后再添加一个滚动视图，并在该视图中添加一个线性布局管理器，同时，还需要在该线性布局管理器中添加一个文本框，用于显示从服务器上读取的微博内容。

（2）在该 MainActivity 中，创建程序中所需的成员变量。其具体代码如下：

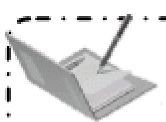
```
private EditText nickname;                          //声明一个输入昵称的编辑框对象
private EditText content;                           //声明一个输入文本内容的编辑框对象
private Button button;                              //声明一个发表按钮对象
private Handler handler;                            //声明一个 Handler 对象
private String result = "";                         //声明一个代表显示内容的字符串
private TextView resultTV;                          //声明一个显示结果的文本框对象
```

（3）编写一个无返回值的 send()方法，用于建立一个 HTTP 连接，并使用 POST 方式将输入的昵称和内容发送到 Web 服务器上，再读取服务器处理的结果。其具体代码如下：

```
public void send() {
    String target = "http://192.168.1.66:8081/blog/dealPost.jsp";    //要提交的目标地址
    URL url;
    try {
        url = new URL(target);
        HttpURLConnection urlConn = (HttpURLConnection) url
                .openConnection();                  //创建一个 HTTP 连接
        urlConn.setRequestMethod("POST");           //指定使用 POST 请求方式
        urlConn.setDoInput(true);                   //向连接中写入数据
```




```
            urlConn.setDoOutput(true);                           //从连接中读取数据
            urlConn.setUseCaches(false);                         //禁止缓存
            urlConn.setInstanceFollowRedirects(true);            //自动执行 HTTP 重定向
            urlConn.setRequestProperty("Content-Type",
                    "application/x-www-form-urlencoded");        //设置内容类型
            DataOutputStream out = new DataOutputStream(
                    urlConn.getOutputStream());                  //获取输出流
            String param = "nickname="
                    + URLEncoder.encode(nickname.getText().toString(), "utf-8")
                    + "&content="
                    + URLEncoder.encode(content.getText().toString(), "utf-8");//要提交的数据
            out.writeBytes(param);                               //将要传递的数据写入数据输出流
            out.flush();                                         //输出缓存
            out.close();                                         //关闭数据输出流
            //判断是否响应成功
            if (urlConn.getResponseCode() == HttpURLConnection.HTTP_OK) {
                InputStreamReader in = new InputStreamReader(
                        urlConn.getInputStream());               //获得读取的内容
                BufferedReader buffer = new BufferedReader(in);  //获取输入流对象
                String inputLine = null;
                while ((inputLine = buffer.readLine()) != null) {
                    result += inputLine + "\n";
                }
                in.close();                                      //关闭字符输入流
            }
            urlConn.disconnect();                                //断开连接
        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

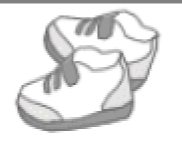


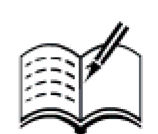
说明：

在设置要提交的数据时，如果包括多个参数，各个参数间使用“&”连接。

（4）在 onCreate()方法中，获取布局管理器中添加的昵称编辑框、内容编辑框、显示结果的文本框和“发表”按钮，并为“发表”按钮添加单击事件监听器。在重写的 onClick()方法中，首先判断输入的昵称和内容是否为空，只要有一个为空，就给出消息提示，否则，创建一个新的线程，用于调用 send()方法发送并读取服务器处理后的微博信息。其具体代码如下：

```
content = (EditText) findViewById(R.id.content);          //获取输入文本内容的 EditText 组件
resultTV = (TextView) findViewById(R.id.result);          //获取显示结果的 TextView 组件
nickname=(EditText)findViewById(R.id.nickname);           //获取输入昵称的 EditText 组件
button = (Button) findViewById(R.id.button);              //获取“发表”按钮组件
//为按钮添加单击事件监听器
button.setOnClickListener(new OnClickListener() {
    @Override
```






```
        public void onClick(View v) {
            if ("".equals(content.getText().toString())) {
                Toast.makeText(MainActivity.this, "请输入要发表的内容！",Toast.LENGTH_SHORT).show();
                return;
            }
            //创建一个新线程，用于发送并读取微博信息
            new Thread(new Runnable() {
                public void run() {
                    send();
                    Message m = handler.obtainMessage();          //获取一个 Message
                    handler.sendMessage(m);                       //发送消息
                }
            }).start();                                           //开启线程
        }
});
```

（5）创建一个 Handler 对象，在重写的 handleMessage()方法中，当变量 result 不为空时，将其显示到结果文本框中，并清空昵称和内容编辑器。其具体代码如下：

```
handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            if (result != null) {
                resultTV.setText(result);                 //显示获得的结果
                content.setText("");                      //清空内容编辑框
                nickname.setText("");                     //清空昵称编辑框
            }
            super.handleMessage(msg);
        }
};
```

（6）由于在本实例中，需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

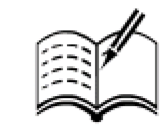
另外，还需要编写一个 Java Web 实例，用于接收 Android 客户端发送的请求，并做出响应，这里编写一个名称为 dealPost.jsp 的文件。在该文件中，首先获取参数 nickname 和 content 指定的昵称和微博信息，并保存到相应的变量中，然后当昵称和微博内容均不为空时，对其进行转码，并获取系统时间，同时组合微博信息输出到页面上。其具体代码如下：

```
<%@ page contentType="text/html; charset=utf-8" language="java" %>
<%
String content=request.getParameter("content");                    //获取输入的微博信息
String nickname=request.getParameter("nickname");                  //获取输入昵称
if(content!=null && nickname!=null){
        nickname=new String(nickname.getBytes("iso-8859-1"),"utf-8");     //对昵称进行转码
        content=new String(content.getBytes("iso-8859-1"),"utf-8");       //对内容进行转码
        String date=new java.util.Date().toLocaleString();                 //获取系统时间
```



```
%>
<%="[ "+nickname+" ]于 "+date+" 发表一条微博，内容如下："%>
<%=content%>
<% }%>
```





将 dealPost.jsp 文件放到 Tomcat 安装路径的 webapps\blog 目录下，并启动 Tomcat 服务器。然后运行本实例，在屏幕上方的编辑框中输入昵称和微博信息，单击“发表”按钮，在下方将显示 Web 服务器的处理结果。例如，输入昵称为“无语”，微博内容为“坚持到底就是胜利！”后，单击“发表”按钮，将显示如图 18.2 所示的运行效果。



图 18.2　应用 POST 方式发表一条微博信息

18.1.2　使用 HttpClient 访问网络

18.1.1 节中介绍了使用 jva.net 包中的 HttpURLConnection 类来访问网络，在一般情况下，如果只需要到某个简单页面提交请求并获取服务器的响应，完全可以使用该技术来实现。不过，对于比较复杂的联网操作，使用 HttpURLConnection 类就不一定能满足要求，这时可以使用 Apache 组织提供的一个 HttpClient 项目来实现。在 Android 中，已经成功地集成了 HttpClient，所以可以直接在 Android 中使用 HttpClient 来访问网络。

HttpClient 实际上是对 Java 提供的访问网络的方法进行了封装。在 HttpURLConnection 类中的输入输出流操作在这个 HttpClient 中被统一封装成了 HttpGet、HttpPost 和 HttpResponse 类，这样，就使操作不那么繁琐。其中，HttpGet 类代表发送 GET 请求；HttpPost 类代表发送 POST 请求；HttpResponse 类代表处理响应的对象。

同使用 HttpURLConnection 类一样，使用该对象发送 HTTP 请求也可以分为 GET 请求和 POST 请求两种，下面分别进行介绍。



1. 发送 GET 请求

同 HttpURLConnection 类一样，使用 HttpClient 发送 GET 请求的方法也比较简单，大致可以分为以下 5 个步骤：

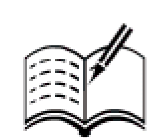
（1）创建 HttpClient 对象。

（2）创建 HttpGet 对象。

（3）如果需要发送请求参数，可以直接将要发送的参数连接到 URL 地址中，也可以调用 HttpGet 的 setParams()方法来添加请求参数。

（4）调用 HttpClient 对象的 execute()方法发送请求。执行该方法将返回一个 HttpResponse 对象。





（5）调用 HttpResponse 的 getEntity()方法，可获得包含服务器响应内容的 HttpEntity 对象，通过该对象可以获取服务器的响应内容。

下面通过一个具体的实例来说明如何使用 HttpClient 来发送 GET 请求。

【例 18.3】 在 Eclipse 中创建 Android 项目，实现使用 HttpClient 向服务器发送 GET 请求，并获取服务器的响应结果。

实例位置：光盘\MR\Instance\18\18.3

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，在默认添加的 TextView 组件的上方添加一个 Button 按钮，并设置其显示文本为“发送 GET 请求”，然后将 TextView 组件的 id 属性修改为 result。

（2）在该 MainActivity 中，创建程序中所需的成员变量。其具体代码如下：

```
private Button button;                                    //声明一个发表按钮对象
private Handler handler;                                  //声明一个 Handler 对象
private String result = "";                               //声明一个代表显示结果的字符串
private TextView resultTV;                                //声明一个显示结果的文本框对象
```

（3）编写一个无返回值的 send()方法，用于建立一个发送 GET 请求的 HTTP 连接，并将指定的参数发送到 Web 服务器上，再读取服务器的响应信息。其具体代码如下：

```
public void send() {
    //要提交的目标地址
    String target = "http://192.168.1.66:8081/blog/deal_httpclient.jsp?param=get";
    HttpClient httpclient = new DefaultHttpClient();         //创建 HttpClient 对象
    HttpGet httpRequest = new HttpGet(target);               //创建 HttpGet 连接对象
    HttpResponse httpResponse;
    try {
        httpResponse = httpclient.execute(httpRequest);      //执行 HttpClient 请求
        if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
            result = EntityUtils.toString(httpResponse.getEntity()); //获取返回的字符串
        }else{
            result="请求失败！";
        }
    } catch (ClientProtocolException e) {
        e.printStackTrace();                                 //输出异常信息
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

（4）在 onCreate()方法中，获取布局管理器中添加的用于显示结果的文本框和“发表”按钮，并为“发表”按钮添加单击事件监听器。在重写的 onClick()方法中，创建并开启一个新的线程，并且在重写的 run()方法中，首先调用 send()方法发送并读取微博信息，然后获取一个 Message 对象，并调用其 sendMessage()方法发送消息。其具体代码如下：

```
resultTV = (TextView) findViewById(R.id.result);                    //获取显示结果的 TextView 组件
button = (Button) findViewById(R.id.button);                        //获取“发表”按钮组件
//为按钮添加单击事件监听器
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {

        //创建一个新线程，用于发送并获取 GET 请求
        new Thread(new Runnable() {
            public void run() {
                send();
                Message m = handler.obtainMessage();    //获取一个 Message
                handler.sendMessage(m);                 //发送消息
            }
        }).start();                                     //开启线程
    }
});
```

（5）创建一个 Handler 对象，在重写的 handleMessage()方法中，当变量 result 不为空时，将其显示到结果文本框中。其具体代码如下：

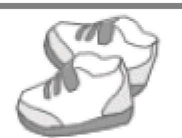
```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (result != null) {
            resultTV.setText(result);                   //显示获得的结果
        }
        super.handleMessage(msg);
    }
};
```

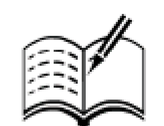
（6）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

另外，还需要编写一个 Java Web 实例，用于接收 Android 客户端发送的请求，并做出响应，这里编写一个名称为 deal_httpclient.jsp 的文件。在该文件中，首先获取参数 param 的值，如果该值不为空，则判断其值是否为 get，如果是 get，则输出文字“发送 GET 请求成功！”。其具体代码如下：

```
<%@ page contentType="text/html; charset=utf-8" language="java" %>
<%
 String param=request.getParameter("param");                //获取参数值
 if(!"".equals(param) || param!=null){
    if("get".equals(param)){
        out.println("发送 GET 请求成功！ ");
    }
```





Note

```
}
%>
```

将 deal_httpclient.jsp 文件放到 Tomcat 安装路径的 webapps\blog 目录下，并启动 Tomcat 服务器。然后运行本实例，单击“发送 GET 请求”按钮，在下方将显示 Web 服务器的处理结果。如果请求发送成功，则显示如图 18.3 所示的运行效果，否则显示文字为“请求失败！”。

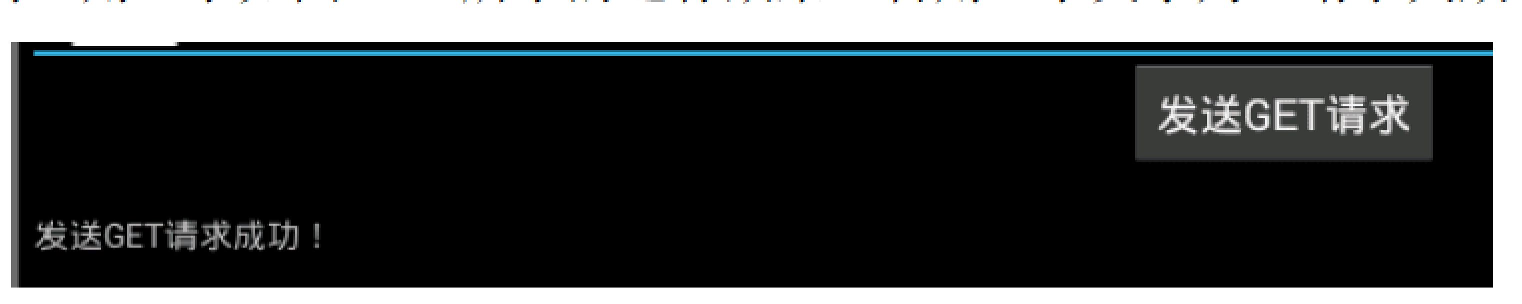


图 18.3　应用 HttpClient 发送 GET 请求

2. 发送 POST 请求

与使用 HttpURLConnection 类发送请求一样，对于复杂的请求数据也需要使用 POST 方式发送。使用 HttpClient 发送 POST 请求大致可以分为以下 5 个步骤。

（1）创建 HttpClient 对象。

（2）创建 HttpPost 对象。

（3）如果需要发送请求参数，可以调用 HttpPost 的 setParams()方法来添加请求参数，也可以调用 setEntity()方法来设置请求参数。

（4）调用 HttpClient 对象的 execute()方法发送请求。执行该方法将返回一个 HttpResponse 对象。

（5）调用 HttpResponse 的 getEntity()方法，可获得包含了服务器响应内容的 HttpEntity 对象，通过该对象可以获取服务器的响应内容。

下面将通过一个具体的实例来说明如何使用 HttpClient 来发送 POST 请求。

【例 18.4】 在 Eclipse 中创建 Android 项目，实现应用 HttpClient 向服务器发送 POST 请求，并获取服务器的响应结果。

实例位置：光盘\MR\Instance\18\18.4

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后在默认添加的线性布局管理器中添加一个 id 为 content 的编辑框（用于输入微博内容），以及一个“发表”按钮，再添加一个滚动视图，并在该视图中添加一个线性布局管理器，最后还需要在该线性布局管理器中添加一个文本框，用于显示从服务器上读取的微博内容。

（2）在该 MainActivity 中，创建程序中所需的成员变量。其具体代码如下：

```
private EditText nickname;            //声明一个输入昵称的编辑框对象
private EditText content;             //声明一个输入文本内容的编辑框对象
private Button button;                //声明一个发表按钮对象
private Handler handler;              //声明一个 Handler 对象
private String result = "";           //声明一个代表显示内容的字符串
private TextView resultTV;            //声明一个显示结果的文本框对象
```

（3）编写一个无返回值的 send()方法，用于建立一个使用 POST 请求方式的 HTTP 连接，



Note

并将输入的昵称和微博内容发送到 Web 服务器上，再读取服务器处理的结果。其具体代码如下：

```
public void send() {
	//要提交的目标地址
	String target = "http://192.168.1.66:8081/blog/deal_httpclient.jsp";
	HttpClient httpclient = new DefaultHttpClient();			//创建 HttpClient 对象
	HttpPost httpRequest = new HttpPost(target);				//创建 HttpPost 对象
	//将要传递的参数保存到 List 集合中
	List<NameValuePair> params = new ArrayList<NameValuePair>();
	params.add(new BasicNameValuePair("param", "post"));		//标记参数
	params.add(new BasicNameValuePair("nickname", nickname.getText().toString()));//昵称
	params.add(new BasicNameValuePair("content", content.getText().toString())); //内容
	try {
		httpRequest.setEntity(new UrlEncodedFormEntity(params, "utf-8"));	//设置编码方式
		HttpResponse httpResponse = httpclient.execute(httpRequest);//执行 HttpClient 请求
		if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK){
			result += EntityUtils.toString(httpResponse.getEntity());	//获取返回的字符串

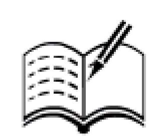
		}else{
			result = "请求失败！";
		}
	} catch (UnsupportedEncodingException e1) {
		e1.printStackTrace();						//输出异常信息
	} catch (ClientProtocolException e) {
		e.printStackTrace();						//输出异常信息
	} catch (IOException e) {
		e.printStackTrace();						//输出异常信息
	}
}
```

（4）在 onCreate()方法中，获取布局管理器中添加的昵称编辑框、内容编辑框、显示结果的文本框和“发表”按钮，并为“发表”按钮添加单击事件监听器。在重写的 onClick()方法中，首先判断输入的昵称和内容是否为空，只要有一个为空，就给出消息提示，否则，创建一个新的线程，调用 send()方法发送并读取服务器处理后的微博信息。其具体代码如下：

```
content = (EditText) findViewById(R.id.content);			//获取输入文本内容的 EditText 组件
resultTV = (TextView) findViewById(R.id.result);			//获取显示结果的 TextView 组件
nickname=(EditText)findViewById(R.id.nickname);			//获取输入昵称的 EditText 组件
button = (Button) findViewById(R.id.button);				//获取“发表”按钮组件
//为按钮添加单击事件监听器
button.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		if ("".equals(content.getText().toString())) {
			Toast.makeText(MainActivity.this, "请输入要发表的内容！",Toast.LENGTH_SHORT).show();
			return;
		}
		//创建一个新线程，用于发送并读取微博信息
		new Thread(new Runnable() {
			public void run() {
```



```
                send();
                Message m = handler.obtainMessage();        //获取一个 Message
                handler.sendMessage(m);                     //发送消息
            }
        }).start();                                         //开启线程
    }
});
```



Note

（5）创建一个 Handler 对象，在重写的 handleMessage()方法中，当变量 result 不为空时，将其显示到结果文本框中，并清空昵称和内容编辑器。其具体代码如下：

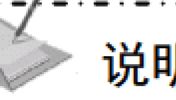
```
handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        if (result != null) {
            resultTV.setText(result);                       //显示获得的结果
            content.setText("");                            //清空内容编辑框
            nickname.setText("");                           //清空昵称编辑框
        }
        super.handleMessage(msg);
    }
};
```

（6）由于在本实例中，需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

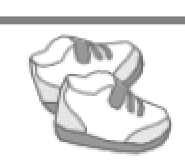
另外，还需要编写一个 Java Web 实例，用于接收 Android 客户端发送的请求，并做出响应。这里仍然使用例 18.3 中创建的 deal_httpclient.jsp 文件，在该文件的 if 语句的结尾处再添加一个 else if 语句，用于处理当请求参数 param 的值为 post 的情况。其关键代码如下：

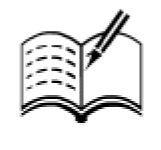
```
else if("post".equals(param)){
    String content=request.getParameter("content");             //获取输入的微博信息
    String nickname=request.getParameter("nickname");           //获取输入昵称
    if(content!=null && nickname!=null){
        nickname=new String(nickname.getBytes("iso-8859-1"),"utf-8");//对昵称进行转码
        content=new String(content.getBytes("iso-8859-1"),"utf-8");  //对内容进行转码
        String date=new java.util.Date().toLocaleString();           //获取系统时间
        out.println("[ "+nickname+" ]于 "+date+" 发表一条微博，内容如下：");
        out.println(content);
    }
}
```



说明：

上面的代码中，首先获取参数 nickname 和 content 指定的昵称和微博信息，并保存到相应的变量中，然后当昵称和微博内容均不为空时，对其进行转码，并获取系统时间，同时组合微博信息输出到页面上。





Note

将 deal_httpclient.jsp 文件放到 Tomcat 安装路径的 webapps\blog 目录下，并启动 Tomcat 服务器。然后运行本实例，在屏幕上方的编辑框中输入昵称和微博信息，单击“发表”按钮，在下方将显示 Web 服务器的处理结果。实例的运行效果如图 18.4 所示。



图 18.4　应用 HttpClient 发送 POST 请求

18.2　使用 WebView 显示网页

Android 提供了内置的浏览器，该浏览器使用了开源的 WebKit 引擎。WebKit 不仅能够搜索网址、查看电子邮件，而且能够播放视频节目。在 Android 中要使用这个内置的浏览器需要通过 WebView 组件来实现。通过 WebView 组件可以轻松地实现显示网页功能。下面对如何使用 WebView 组件显示网页进行详细介绍。

18.2.1　使用 WebView 组件浏览网页

WebView 组件是专门用来浏览网页的，它的使用方法与其他组件一样，既可以在 XML 布局文件中使用<WebView>标记添加，又可以在 Java 文件中通过 new 关键字创建出来。推荐采用第一种方法，也就是通过<WebView>标记在 XML 布局文件中添加。在 XML 布局文件中添加一个 WebView 组件可以使用下面的代码。

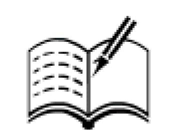
```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

添加 WebView 组件后，就可以应用该组件提供的方法来执行浏览器操作。Web 组件提供的常用方法如表 18.2 所示。

表 18.2　WebView 组件提供的常用方法

方　法	描　述
loadUrl(String url)	用于加载指定 URL 对应的网页
loadData(String data, String mimeType, String encoding)	用于将指定的字符串数据加载到浏览器中
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl)	用于基于 URL 加载指定的数据
capturePicture()	用于创建当前屏幕的快照





续表

方　法	描　述
goBack()	执行后退操作，相当于浏览器上的后退按钮的功能
goForward()	执行前进操作，相当于浏览器上的前进按钮的功能
stopLoading()	用于停止加载当前页面
reload()	用于刷新当前页面

下面将通过一个具体的例子来说明如何使用 WebView 组件浏览网页。

【例 18.5】 在 Eclipse 中创建 Android 项目，实现应用 WebView 组件浏览指定网页。

实例位置：光盘\MR\Instance\18\18.5

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个 WebView 组件。其关键代码如下：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（2）在 MainActivity 的 onCreate()方法中，获取布局管理器中添加的 WebView 组件，并为其指定要加载网页的 URL 地址。其具体代码如下：

```
WebView webview=(WebView)findViewById(R.id.webView1);   //获取布局管理器中添加的 WebView 组件
webview.loadUrl("http://192.168.1.66:8081/bbs/");       //指定要加载的网页
```

（3）由于在本实例中，需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网络资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

运行本实例，在屏幕上将显示通过 URL 地址指定的网页，如图 18.5 所示。



图 18.5　使用 WebView 浏览网页

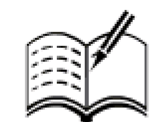




技巧：

如果想让 WebView 组件具有放大和缩小网页的功能，需要进行以下设置。

```
webview.getSettings().setSupportZoom(true);
webview.getSettings().setBuiltInZoomControls(true);
```


Note

18.2.2 使用 WebView 组件加载 HTML 代码

在进行 Android 开发时，对于一些游戏的帮助信息，使用 HTML 代码进行显示比较实用，这样不仅界面更加美观，而且可以让开发更加简单、快捷。WebView 组件提供了 loadData()和 loadDataWithBaseURL()方法来加载 HTML 代码。但是，使用 loadData()方法加载带中文的 HTML 内容时，会产生乱码，使用 loadDataWithBaseURL()方法就不会出现中文乱码的情况。loadDataWithBaseURL()方法的基本语法格式如下：

```
loadDataWithBaseURL(String baseUrl, String data, String mimeType, String encoding, String historyUrl)
```

loadDataWithBaseURL()方法的各参数说明如表 18.3 所示。

表 18.3　loadDataWithBaseURL()方法的参数说明

参　数	描　述
baseUrl	用于指定当前页使用的基本 URL。如果为 null，则使用默认的 about:blank，也就是空白页
data	用于指定要显示的字符串数据
mimeType	用于指定要显示内容的 MIME 类型。如果为 null，默认使用 text/html
encoding	用于指定数据的编码方式
historyUrl	用于指定当前页的历史 URL，也就是进入该页前显示页的 URL。如果为 null，则使用默认的 about:blank

下面将通过一个具体的实例来说明如何使用 WebView 组件加载 HTML 代码。

【例 18.6】 在 Eclipse 中创建 Android 项目，实现应用 WebView 组件加载使用 HTML 代码添加的帮助信息。

实例位置：光盘\MR\Instance\18\18.6

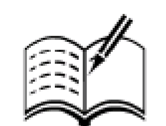
程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个 WebView 组件。其关键代码如下：

```
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（2）在 MainActivity 的 onCreate()方法中，首先获取布局管理器中添加的 WebView 组件，





然后创建一个字符串构建器，将要显示的 HTML 代码放置在该构建器中，最后再应用 loadDataWithBaseURL()方法加载构建器中的 HTML 代码。其具体代码如下：

```
WebView webview=(WebView)findViewById(R.id.webView1);   //获取布局管理器中添加的 WebView 组件
StringBuilder sb=new StringBuilder();//创建一个字符串构建器，将要显示的 HTML 内容放置在该构建器中
sb.append("<div>选择选项，然后从以下选项中进行选择：</div>");
sb.append("<ul>");
sb.append("<li>编辑内容：用于增加、移动和删除桌面上的快捷工具。</li>");
sb.append("<li>隐藏内容：用于隐藏桌面上的小工具。</li>");
sb.append("<li>显示内容：用于显示桌面上的小工具。</li>");
sb.append("</ul>");
webview.loadDataWithBaseURL(null, sb.toString(), "text/html", "utf-8", null);//加载数据
```

运行本实例，在屏幕上将显示如图 18.6 所示的由 HTML 代码指定的帮助信息。

选择选项，然后从以下选项中进行选择：

- 编辑内容：用于增加、移动和删除桌面上的快捷工具。
- 隐藏内容：用于隐藏桌面上的小工具。
- 显示内容：用于显示桌面上的小工具。

图 18.6　使用 WebView 加载 HTML 代码

18.2.3　让 WebView 组件支持 JavaScript

在默认情况下，WebView 组件是不支持 JavaScript 的，但是在运行某些不得不使用 JavaScript 代码的网站时，还需要让它支持 JavaScript。实际上，让 WebView 组件支持 JavaScript 也比较简单，只需以下两个步骤就可以实现。

（1）使用 WebView 组件的 WebSettings 对象提供的 setJavaScriptEnabled()方法让 JavaScript 可用。例如，存在一个名称为 webview 的 WebView 组件，要设置在该组件中允许使用 JavaScript，可以使用下面的代码。

```
webview.getSettings().setJavaScriptEnabled(true);        //设置 JavaScript 可用
```

（2）经过以上设置后，网页中的大部分 JavaScript 代码均可用。但是，对于通过 window.alert()方法弹出的对话框并不可用。要想显示弹出的对话框，需要使用 WebView 组件的 setWebChromeClient()方法来处理 JavaScript 的对话框。其具体代码如下：

```
webview.setWebChromeClient(new WebChromeClient());
```

这样设置后，在使用 WebView 组件显示带弹出 JavaScript 对话框的网页时，网页中弹出的对话框将不会被屏蔽掉。下面将通过一个具体的实例来说如何让 WebView 组件支持 JavaScript。

【例 18.7】 在 Eclipse 中创建 Android 项目，实现控制 WebView 组件是否允许 JavaScrpt。

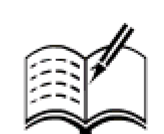
实例位置：光盘\MR\Instance\18\18.7

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件



删除，然后添加一个 CheckBox 和一个 WebView 组件。其关键代码如下：



Note

```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="允许执行 JavaScript 代码" />
<WebView
    android:id="@+id/webView1"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

（2）在 MainActivity 中，声明一个 WebView 组件的对象 webview。其具体代码如下：

```
private WebView webview;                                //声明 WebView 组件的对象
```

（3）在 onCreate()方法中，首先获取布局管理器中添加的 WebView 组件和复选框组件，然后为复选框组件添加选中状态被改变的事件监听器。在重写的 onCheckedChanged()方法中，根据复选框的选中状态决定是否允许使用 JavaScript，最后再为 WebView 组件指定要加载的网页。其具体代码如下：

```
webview = (WebView) findViewById(R.id.webView1);          //获取布局管理器中添加的 WebView 组件
CheckBox check = (CheckBox) findViewById(R.id.checkBox1); //获取布局管理器中添加的复选框组件
check.setOnCheckedChangeListener(new OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
        if (isChecked) {
            webview.getSettings().setJavaScriptEnabled(true); //设置 JavaScript 可用
            webview.setWebChromeClient(new WebChromeClient());
            webview.loadUrl("http://192.168.1.66:8081/bbs/allowJS.jsp");//指定要加载网页
        }else{
            webview.loadUrl("http://192.168.1.66:8081/bbs/allowJS.jsp");//指定要加载网页
        }
    }
});
webview.loadUrl("http://192.168.1.66:8081/bbs/allowJS.jsp");   //指定要加载网页
```

（4）由于在本实例中需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

运行本实例，在屏幕上将显示不支持 JavaScript 的网页，选中上面的“允许执行 JavaScript 代码”复选框后，该网页将支持 JavaScript。例如，选中“允许执行 JavaScript 代码”复选框后，再单击网页中的“发表”按钮，将弹出一个提示对话框，如图 18.7 所示。





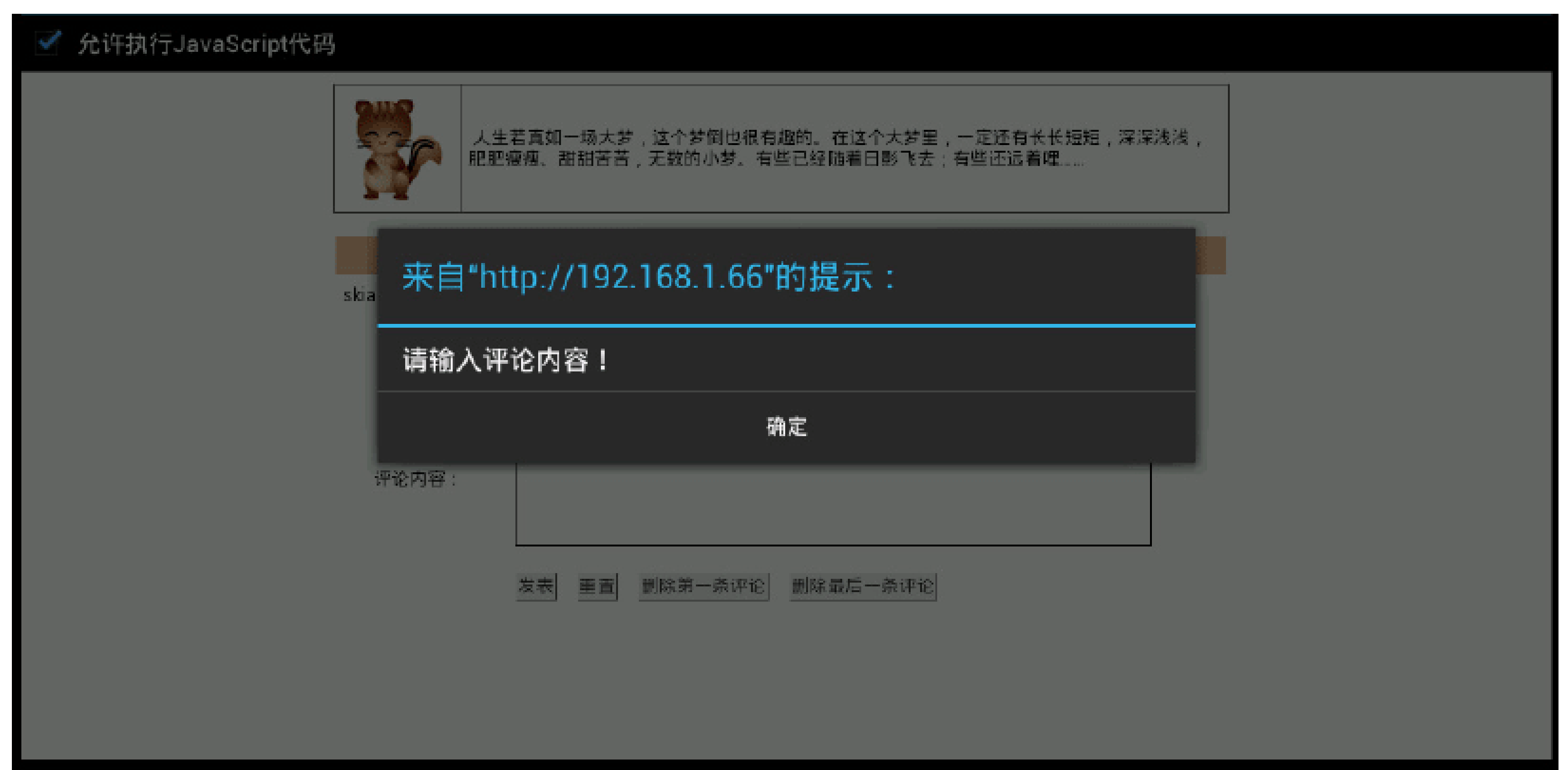




图 18.7　让 WebView 组件允许执行 JavaScript

18.3　综 合 应 用

18.3.1　打造功能实用的网页浏览器

【例 18.8】 在 Android 中制作一个包含前进、后退和支持 JavaScript 的网页浏览器。运行程序，单击 GO 按钮，访问地址栏中指定的网站；单击“前进”和“后退”按钮，实现类似于 IE 浏览器上的前进和后退功能，运行效果如图 18.8 所示。

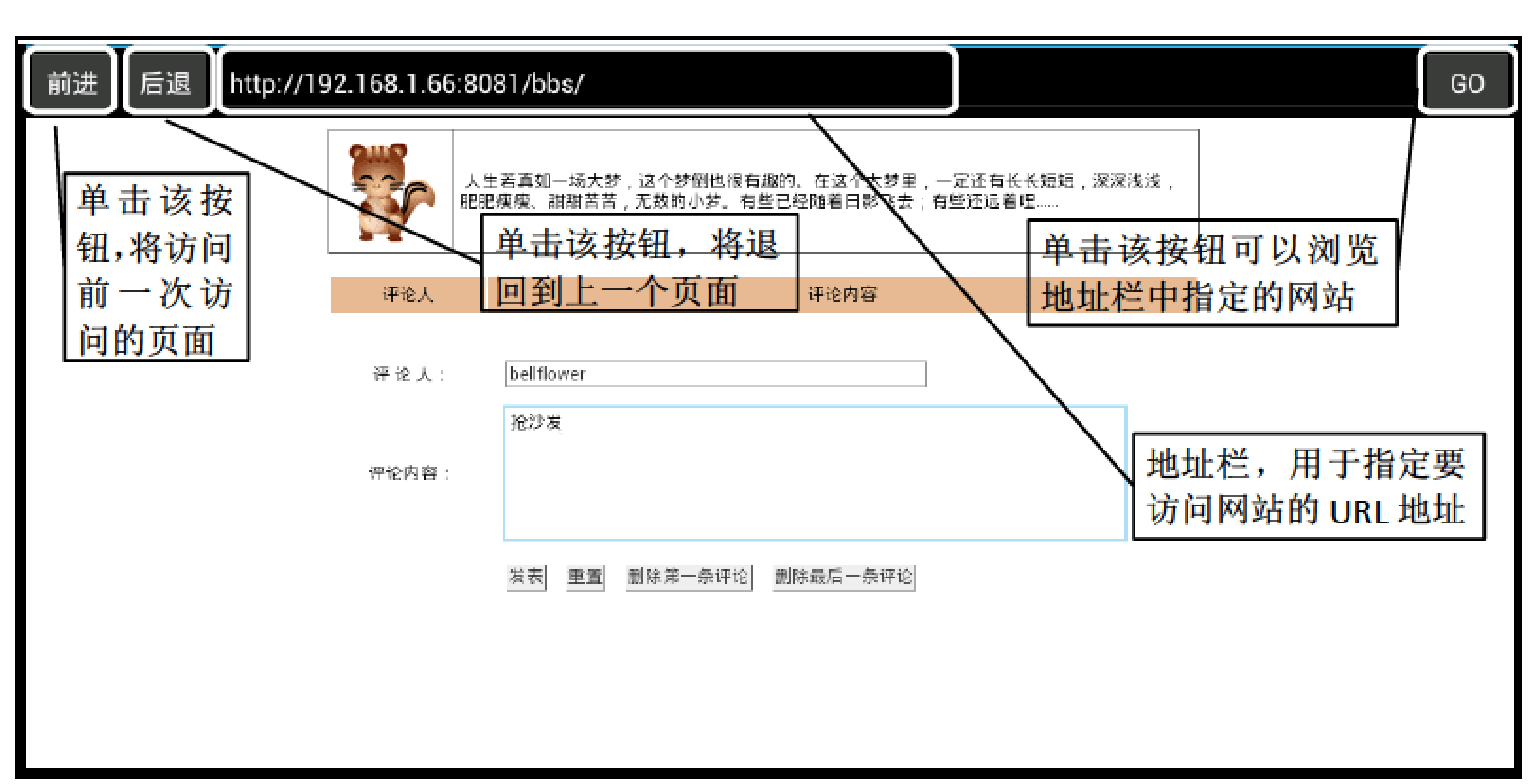
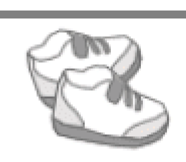


图 18.8　打造功能实用的网页浏览器

实例位置：光盘\MR\Instance\18\18.8

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个水平的线性布局管理器和一个用于显示网页的 WebView 组件，并在该布局管理器中添加“前进”按钮、“后退”按钮、地址栏编辑框和 GO 按钮。



（2）在 MainActivity 中，声明一个 WebView 组件的对象 webView。其具体代码如下：

```
private WebView webView;                                    //声明 WebView 组件的对象
private EditText urlText;                                   //声明作为地址栏的 EditText 对象
private Button goButton;                                    //声明 GO 按钮对象
```

Note

（3）在 onCreate()方法中，首先获取布局管理器中添加的作为地址栏的 EditText 组件、GO 按钮和 WebView 组件，然后让 WebView 组件支持 JavaScript，以及为 WebView 组件设置处理各种通知和请求事件。其具体代码如下：

```
urlText=(EditText)findViewById(R.id.editText_url);          //获取布局管理器中添加的地址栏
goButton=(Button)findViewById(R.id.button_go);              //获取布局管理器中添加的 GO 按钮
webView=(WebView)findViewById(R.id.webView1);               //获取 WebView 组件
webView.getSettings().setJavaScriptEnabled(true);           //设置 JavaScript 可用
webView.setWebChromeClient(new WebChromeClient());          //处理 JavaScript 对话框
//处理各种通知和请求事件，如果不使用该句代码，将使用内置浏览器访问网页
webView.setWebViewClient(new WebViewClient());
```

注意：

上面的代码中，加粗的代码一定不能省略，如果不使用该句代码，将使用内置浏览器访问网页。

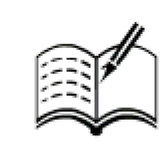
（4）获取布局管理器中添加的“前进”和“后退”按钮，并分别为它们添加单击事件监听器，在“前进”按钮的 onClick()方法中调用 goForward()方法实现前进功能；在“后退”按钮的 onClick()方法中调用 goBack()方法实现后退功能。其具体代码如下：

```
Button forward=(Button)findViewById(R.id.forward);          //获取布局管理器中添加的“前进”按钮
forward.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        webView.goForward();                                //前进
    }
});
Button back=(Button)findViewById(R.id.back);                //获取布局管理器中添加的“后退”按钮
back.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        webView.goBack();                                   //后退
    }
});
```

（5）为地址栏添加键盘按键被按下的事件监听器，实现当按下键盘上的回车键时，如果地址栏中的 URL 地址不为空，则调用 openBrowser()方法浏览网页，否则调用 showDialog()方法弹出提示对话框。其具体代码如下：

```
urlText.setOnKeyListener(new OnKeyListener() {
    @Override
```






```
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            if(keyCode==KeyEvent.KEYCODE_ENTER){                //如果为回车键
                if(!"".equals(urlText.getText().toString())){
                        openBrowser();                          //浏览网页
                        return true;
                }else{
                    showDialog();                               //弹出提示对话框
                }
            }
            return false;
        }
});
```

（6）为 GO 按钮添加单击事件监听器，实现单击该按钮时，如果地址栏中的 URL 地址不为空，则调用 openBrowser()方法浏览网页，否则调用 showDialog()方法弹出提示对话框。其具体代码如下：

```
goButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if(!"".equals(urlText.getText().toString())){
                openBrowser();                                  //浏览网页
            }else{
                showDialog();                                   //弹出提示对话框
            }
        }
});
```

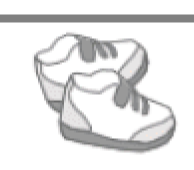
（7）编写 openBrowser()方法，用于浏览网页。其具体代码如下：

```
private void openBrowser(){
        webView.loadUrl(urlText.getText().toString());          //浏览网页
        Toast.makeText(this, "正在加载："+urlText.getText().toString(), Toast.LENGTH_SHORT).show();
}
```

（8）编写 showDialog()方法，用于显示一个带“确定”按钮的对话框，通知用户需要输入要访问的网址。showDialog()方法的具体代码如下：

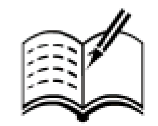
```
private void showDialog(){
        new AlertDialog.Builder(MainActivity.this)
        .setTitle("网页浏览器")
        .setMessage("请输入要访问的网址")
        .setPositiveButton("确定",new DialogInterface.OnClickListener(){
            public void onClick(DialogInterface dialog,int which){
                Log.d("WebWiew","单击确定按钮");
            }
        }).show();
}
```

（9）由于在本实例中，需要访问网络资源，所以还需要在 AndroidManifest.xml 文件中指定



允许访问网格资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```



Note

18.3.2 获取天气预报

【例 18.9】 本实例主要实现获取指定城市天气预报的功能，运行程序，在屏幕上将显示默认城市的天气预报信息，单击上方的“北京”、“上海”、“哈尔滨”、“长春”、“沈阳”和“广州”等按钮，将显示对应城市的天气预报信息。例如，单击“长春”按钮，将显示如图 18.9 所示的效果。



图 18.9　获取长春市的天气预报

实例位置：光盘\MR\Instance\18\18.9

程序的开发步骤如下：

（1）修改新建项目的 res\layout 目录下的布局文件 main.xml，将默认添加的 TextView 组件删除，然后添加一个水平的线性布局管理器和一个用于显示网页的 WebView 组件，并在该布局管理器中添加“北京”、“上海”、“哈尔滨”、“长春”、“沈阳”和“广州”按钮。

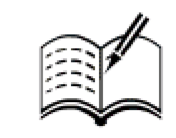
（2）在 MainActivity 中，声明一个 WebView 组件的对象 webView。其具体代码如下：

```
private WebView webView;                                    //声明 WebView 组件的对象
```

（3）在 onCreate()方法中，首先获取布局管理器中添加的 WebView 组件，然后设置该组件允许使用 JavaScript，以及处理 JavaScript 对话框和各种请求事件，再为 WebView 组件指定要加载的天气预报信息，最后将网页内容放大 4 倍。其具体代码如下：

```
webView=(WebView)findViewById(R.id.webView1);               //获取 WebView 组件
webView.getSettings().setJavaScriptEnabled(true);           //设置 JavaScript 可用
webView.setWebChromeClient(new WebChromeClient());          //处理 JavaScript 对话框
//处理各种通知和请求事件，如果不使用该句代码，将使用内置浏览器访问网页
webView.setWebViewClient(new WebViewClient());
webView.loadUrl("http://m.weather.com.cn/m/pn12/weather.htm ");//设置默认显示的天气预报信息
webView.setInitialScale(57*4);                              //将网页内容放大 4 倍
```





（4）让 MainActivity 实现 OnClickListener 接口，用于添加单击事件监听器。其代码如下：

```
public class MainActivity extends Activity implements OnClickListener {
```

（5）重写 onClick()方法，用于为屏幕中的各个按钮的单击事件设置不同的响应，也就是在单击各个按钮时，调用 openUrl()方法获取不同地区的天气预报信息。其具体代码如下：

```
@Override
public void onClick(View view){
    switch(view.getId()){
    case R.id.bj:        //单击的是“北京”按钮
        openUrl("101010100T");
        break;
    case R.id.sh:        //单击的是“上海”按钮
        openUrl("101020100T");
        break;
    case R.id.heb:       //单击的是“哈尔滨”按钮
        openUrl("101050101T");
        break;
    case R.id.cc:        //单击的是“长春”按钮
        openUrl("101060101T");
        break;
    case R.id.sy:        //单击的是“沈阳”按钮
        openUrl("101070101T");
        break;
    case R.id.gz:        //单击的是“广州”按钮
        openUrl("101280101T");
        break;
    }
}
```

（6）获取布局管理器中添加的“北京”、“上海”、“哈尔滨”、“长春”、“沈阳”和“广州”按钮，并分别为它们添加单击事件监听器。其具体代码如下：

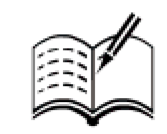
```
Button bj=(Button)findViewById(R.id.bj);        //获取布局管理器中添加的“北京”按钮
bj.setOnClickListener(this);
Button sh=(Button)findViewById(R.id.sh);        //获取布局管理器中添加的“上海”按钮
sh.setOnClickListener(this);
Button heb=(Button)findViewById(R.id.heb);      //获取布局管理器中添加的“哈尔滨”按钮
heb.setOnClickListener(this);
Button cc=(Button)findViewById(R.id.cc);        //获取布局管理器中添加的“长春”按钮
cc.setOnClickListener(this);
Button sy=(Button)findViewById(R.id.sy);        //获取布局管理器中添加的“沈阳”按钮
sy.setOnClickListener(this);
Button gz=(Button)findViewById(R.id.gz);        //获取布局管理器中添加的“广州”按钮
gz.setOnClickListener(this);
```

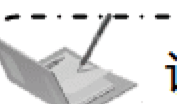
（7）编写用于打开网页获取天气预报信息的方法 openUrl()，在该方法中，将根据传递的参



数不同，获取不同地区的天气预报信息。其具体代码如下：

```
private void openUrl(String id){
    //获取并显示天气预报信息
    webView.loadUrl("http://m.weather.com.cn/m/pn12/weather.htm?id="+id+" ");
}
```


Note


说明：

在中国天气网（http://www.weather.com.cn/）中提供了单城市24小时天气预报插件，使用这个插件可以实现在Android中获取指定城市的天气预报。

（8）由于在本实例中，需要访问网络资源，所以还需要在AndroidManifest.xml文件中指定允许访问网络资源的权限。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```

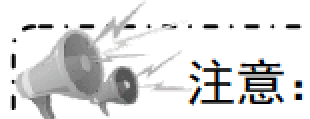
18.4　本章常见错误

运行Android网络应用时，出现下面的错误提示：

```
Permission denied(maybe missing internet permission)
```

该错误是由于用户没有网络访问权限造成的。解决该错误，只需要在AndroidManifest.xml文件中添加允许使用网络选项的权限即可。其具体代码如下：

```
<uses-permission android:name="android.permission.INTERNET"/>
```


注意：

上面的代码需要编写在</application>结束标签的后面。

18.5　本 章 小 结

本章首先介绍了通过HTTP访问网络主要有两种方法：一种是使用java.net包中的HttpURLConnection实现，另一种是通过Android提供的HttpClient实现。对于一些简单的访问网络的操作可以使用HttpURLConnection实现，但是如果是比较复杂的操作，就需要使用HttpClient来实现了。在介绍了通过HTTP访问网络以后，又介绍了使用Android提供的WebView组件来显示网页，使用该组件可以很方便地实现基本的网页浏览器功能。





18.6　跟我上机

参考答案：光盘\MR\跟我上机

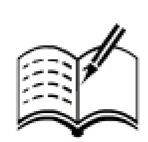
开发一个 Android 程序，主要实现从指定网站下载文件的功能。将程序布局文件中默认添加的 LinearLayout 布局管理器修改为水平布局管理器，并将默认添加的 TextView 组件设置 android:id 属性为@+id/editText_url，android:layout_weight 属性为 1，android:text 属性为@string/defaultvalue，android:lines 属性为 1；然后在 TextView 组件的右侧添加一个“下载”按钮。布局界面如图 18.10 所示。



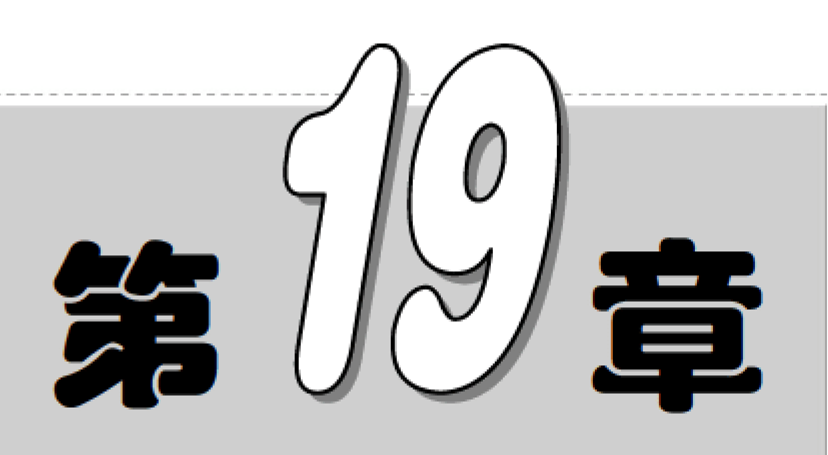
图 18.10　从指定网站下载文件

下载文件的功能是在“下载”按钮中实现的，为“下载”按钮添加单击事件监听器，在重写的 onClick()方法中，创建并开启一个新线程，用于从网络上获取文件，在重写的 run()方法中，首先获取文件的下载地址，并创建一个相关的链接，然后获取输入流对象，并从下载地址中获取到要下载文件的文件名及扩展名，再读取文件到一个输出流对象中，并关闭相关对象及断开连接，最后获取一个 Message 并发送消息。下载文件功能的实现代码参考如下：

```
button = (Button) findViewById(R.id.button_go);                //获取布局管理器中添加的“下载”按钮
//为“下载”按钮添加单击事件监听器
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        //创建一个新线程，用于从网络上获取文件
        new Thread(new Runnable() {
            public void run() {
                try {
                    String sourceUrl = urlText.getText().toString();    //获取下载地址
                    URL url = new URL(sourceUrl);          //创建下载地址对应的 URL 对象
```


Note

```
                    HttpURLConnection urlConn = (HttpURLConnection) url
                            .openConnection();                          //创建一个连接
                    InputStream is = urlConn.getInputStream();          //获取输入流对象
                    if (is != null) {
                        String expandName = sourceUrl.substring(
                                sourceUrl.lastIndexOf(".") + 1,
                                sourceUrl.length()).toLowerCase();      //获取文件的扩展名
                        String fileName = sourceUrl.substring(
                                sourceUrl.lastIndexOf("/") + 1,
                                sourceUrl.lastIndexOf("."));            //获取文件名
                        File file = new File("/sdcard/pictures/"
                                + fileName + "." + expandName);         //在 SD 卡上创建文件
                        FileOutputStream fos = new FileOutputStream(
                                file);                                  //创建文件输出流对象
                        byte buf[] = new byte[128];                     //创建 1B 数组
                        //读取文件到输出流对象中
                        while (true) {
                            int numread = is.read(buf);
                            if (numread <= 0) {
                                break;
                            } else {
                                fos.write(buf, 0, numread);
                            }
                        }
                    }
                    is.close();                                         //关闭输入流对象
                    urlConn.disconnect();                               //关闭连接
                    flag = true;
                } catch (MalformedURLException e) {
                    e.printStackTrace();                                //输出异常信息
                    flag = false;
                } catch (IOException e) {
                    e.printStackTrace();                                //输出异常信息
                    flag = false;
                }
                Message m = handler.obtainMessage();                    //获取一个 Message
                handler.sendMessage(m);                                 //发送消息
            }
        }).start();                                                     //开启线程
    }
});
```



Service服务的使用

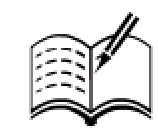
（视频讲解：40分钟）

Service 用于在后台完成用户指定的操作，它可以用于音乐播放器、文件下载工具等应用程序。用户可以使用其他控件来与 Service 进行通信。本章将详细讲解 Service 服务的使用。

本章能够完成的主要范例（已掌握的在方框中打勾）

- ☐ 继承 IntentService 输出当前时间
- ☐ 继承 Service 输出当前时间
- ☐ 继承 Binder 类绑定服务显示时间
- ☐ 使用 Messenger 类绑定服务显示时间
- ☐ 视力保护程序





Note

19.1 Service 概述

Service（服务）是能够在后台执行长时间运行操作并且不提供用户界面的应用程序组件，其他应用程序组件能启动服务并且即便用户切换到另一个应用程序，服务还是可以在后台运行。此外，组件能够绑定到服务并与之交互，甚至执行进程间通信（IPC）。例如，服务能在后台处理网络事务、播放音乐、执行文件 I/O 或者与 ContentProvider 通信。

19.1.1 Service 的分类

服务从本质上可以分为以下两种类型。

☑ Started（启动）：当应用程序组件（如 Activity）通过调用 startService()方法启动服务时，服务处于 started 状态。一旦启动，服务能在后台无限期运行，即使启动它的组件已经被销毁。通常，启动服务执行单个操作并且不会向调用者返回结果。例如，它可能通过网络下载或者上传文件。如果操作完成，服务需要停止自身。

☑ Bound（绑定）：当应用程序组件通过调用 bindService()方法绑定到服务时，服务处于 bound 状态。绑定服务提供客户端-服务器接口以允许组件与服务交互、发送请求、获得结果、甚至使用进程间通信（IPC）跨进程完成这些操作。仅当其他应用程序组件与之绑定时，绑定服务才运行。多个组件可以一次绑定到一个服务上，但是当它们都解绑定时，服务被销毁。

尽管本章将两种类型的服务分开讨论，服务也可以同时属于两种类型，它可以启动（无限期运行）也能绑定。其重点在于是否实现一些回调方法：onStartCommand()方法允许组件启动服务，onBind()方法允许组件绑定服务。

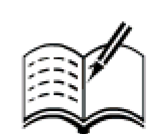
不管应用程序是否为启动状态、绑定状态或者两者，都能通过 Intent 使用服务（甚至从独立的应用程序），就像使用 Activity 那样。然而，开发人员可以在配置文件中将服务声明为私有的，从而阻止其他应用程序访问。

服务运行于管理它的进程的主线程，服务不会创建自己的线程也不会运行于独立的进程（除非开发人员定义）。这意味着，如果服务将完成 CPU 密集工作或者阻塞操作（如 MP3 回放或者联网），开发人员需要在服务中创建新线程来完成这些工作。通过使用独立的线程，开发人员能减少应用程序不响应（ANR）错误的风险并且应用程序主线程仍然能用于用户与 Activity 交互。

19.1.2 Service 类的重要方法

为了创建服务，开发人员需要创建 Service 类（或其子类）的子类。在实现类中，需要重写一些处理服务生命周期重要方面的回调方法，并根据需要提供组件绑定到服务的机制。需要重写的重要回调方法如下：





☑ onStartCommand()

当其他组件，例如 Activity 调用 startService()方法请求服务启动时，系统调用该方法。一旦该方法执行，服务就启动（处于 started 状态）并在后台无限期运行。如果开发人员实现该方法，则需要在任务完成时调用 stopSelf()或 stopService()方法停止服务（如果仅想提供绑定，则不必实现该方法）。

☑ onBind()

当其他组件调用 bindService()方法想与服务绑定时（如执行 RPC），系统调用该方法。在该方法的实现中，开发人员必须通过返回 IBinder 提供客户端用来与服务通信的接口。该方法必须实现，但是如果不想允许绑定，则应该返回 null。

☑ onCreate()

当服务第一次创建时，系统调用该方法执行一次性建立过程（在系统调用 onStartCommand()或 onBind()方法前）。如果服务已经运行，该方法不被调用。

☑ onDestroy()

当服务不再使用并即将销毁时，系统调用该方法。服务应该实现该方法来清理如线程、注册监听器和接收者等资源。这是服务收到的最后调用。

如果组件调用 startService()方法启动服务（onStartCommand()方法被调用），服务需要使用 stopSelf()方法停止自身，或者其他组件使用 stopService()方法停止该服务。

如果组件调用 bindService()方法创建服务（onStartCommand()方法不被调用），服务运行时间与组件绑定到服务的时间一样长。一旦服务从所有客户端解绑定，系统会将其销毁。

Android 系统仅当内存不足并且必须回收系统资源来显示用户关注的 Activity 时，才会强制停止服务。如果服务绑定到用户关注的 Activity，则会降低停止概率。如果服务被声明为前台运行，则基本不会停止。否则，如果服务是 started 状态并且长时间运行，则系统会随时间推移降低其在后台任务列表中的位置并且服务有很大概率被停止。如果服务是 started 状态，则必须设计如何优雅地重启服务。如果系统停止服务，则资源可用时就会重启它（尽管这也依赖于 onStartCommand()方法的返回值）。

Service 类的继承关系如图 19.1 所示。

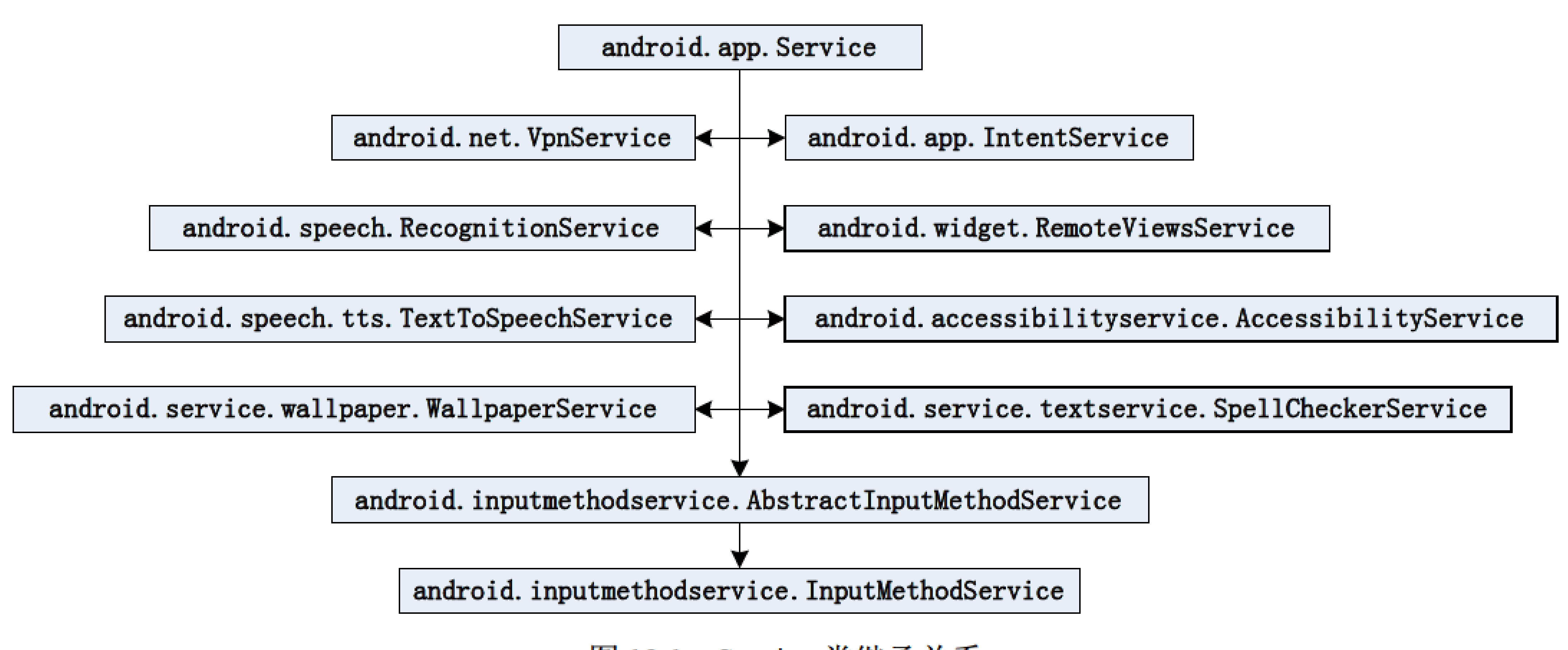
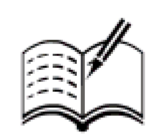


图 19.1　Service 类继承关系



19.1.3　Service 的声明



Note

类似 Activity 和其他组件，开发人员必须在应用程序配置文件中声明全部的 Service。为了声明 Service，需要向<application>标记中增加<service>子标记。<service>子标记的语法如下：

```
<service android:enabled=["true" | "false"]
        android:exported=["true" | "false"]
        android:icon="drawable resource"
        android:label="string resource"
        android:name="string"
        android:permission="string"
        android:process="string" >
        . . .
</service>
```

各个标记属性的说明如下。

☑　android:enabled

服务能否被系统实例化，true 表示可以，false 表示不可用，默认值是 true。<application>标记也有自己的 enabled 属性，用于包括服务的全部应用程序组件。application 和 service 属性必须同时设置成 true（两者的默认值也都是 true）才能让服务可用。如果任何一个是 false，服务被禁用并且不能实例化。

☑　android:exported

其他应用程序组件能否调用服务或者与其交互，true 表示可以，false 表示不可以。当该值是 false 时，只有同一个应用程序的组件或者具有相同用户 ID 的应用程序能启动或者绑定到服务。

默认值依赖于服务是否包含 Intent 过滤器。没有过滤器说明它仅能通过精确类名调用。这意味着服务仅用于应用程序内部使用（因为其他可能不知道类名）。此时，默认值是 false。另一方面，存在至少一个过滤器暗示服务可以用于外部使用，因此默认值是 true。

该属性不是限制其他应用程序使用服务的唯一方式，还可以使用 permission 属性限制外部实体与服务交互。

☑　android:icon

表示服务的图标。该属性必须设置成包含图片定义的可绘制资源引用。如果没有设置，使用应用程序图标取代。

服务图标，不管在此设置还是在<application>标记设置，都是所有服务的 Intent 过滤器默认图标。

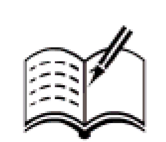
☑　android:label

显示给用户的服务名称。如果没有设置，使用应用程序标签取代。

服务标签，不管在此设置还是在<application>标记设置，都是所有服务的 Intent 过滤器默认图标。

标签应该设置为字符串资源引用，这样它能像用户界面的其他字符串那样本地化。然而，为了开发时方便，也可以设置成原始字符串。





☑ android:name

实现服务的 Service 子类名称。这应该是一个完整的类名（如 com.mingrisoft.RoomService）。然而，为了简便，如果名称的第一个符号是点号（如.RoomService），它会增加在<manifest>标记中定义的包名。

一旦发布了应用程序，不应该再修改这个名称。它没有默认值并且必须指定。

☑ android:permission

实体必须包含的权限名称，以便启动或者绑定到服务。如果 startService()、bindService()或 stopService()方法调用者没有被授权，方法调用无效并且 Intent 对象也不会发送给服务。

如果该属性没有设置，使用<application>标记的 permission 属性设置给服务。如果<application>和<service>标记的 permission 属性都未设置，服务不受权限保护。

☑ android:process

服务运行的进程名称。通常，应用程序的全部组件运行于为应用程序创建的默认进程。它与应用程序包名相同。<application>标记的 process 属性能为全部组件设置一个不同的默认值。但是组件能用自己的 process 属性重写默认值，从而允许应用程序跨越多个进程。

如果分配给该属性的名称以冒号（:）开头，仅属于应用程序的新进程会在需要时创建，服务能在该进程中运行。如果进程名称以小写字母开头，服务会运行在以此为名的全局进程，但需要提供相应的权限。这允许不同应用程序组件共享进程，减少资源使用。

19.2　Started Service 的使用

Started Service（启动服务）是由其他组件调用 startService()方法启动的，这导致服务的 onStartCommand()方法被调用。

当服务是 started 状态时，它的生命周期与启动它的组件无关并且可以在后台无限期运行，即使启动服务的组件已经被销毁。因此，服务需要在完成任务后调用 stopSelf()方法停止，或者由其他组件调用 stopService()方法停止。

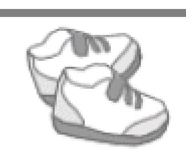
应用程序组件（如 Activity）能通过调用 startService()方法和传递 Intent 对象来启动服务，在 Intent 对象中指定了服务并且包含服务需要使用的全部数据。服务使用 onStartCommand()方法接收 Intent。

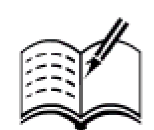
例如，假设 Activity 需要保存一些数据到在线数据库。Activity 可以启动伴侣服务并通过传递 Intent 到 startService()方法来发送需要保存的数据。服务在 onStartCommand()方法中收到 Intent，连入网络并执行数据库事务。当事务完成时，服务停止自身并销毁。

Android 提供了两个类供开发人员继承来创建启动服务。

☑ Service：这是所有服务的基类。当继承该类时，创建新线程来执行服务的全部工作是非常重要的。因为服务默认使用应用程序主线程，这可能降低应用程序 Activity 的运行性能。

☑ IntentService：这是 Service 类的子类，它每次使用一个工作线程来处理全部启动请求。在不必同时处理多个请求时，这是最佳选择。开发人员仅需要实现 onHandleIntent()方法，它接收每次启动请求的 Intent 以便完成后台任务。





19.2.1 继承 IntentService 类

因为多数启动服务不必同时处理多个请求（在多线程情境下会很危险），所以使用 IntentService 类实现服务是非常好的选择。IntentService 完成如下任务。

- ☑ 创建区别于应用程序主线程的默认工作线程来执行发送到 onStartCommand()方法的全部 Intent。
- ☑ 创建工作队列每次传递一个 Intent 到 onHandleIntent()方法实现，这样就不必担心多线程。
- ☑ 所有启动请求处理完毕后停止服务，这样就不必调用 stopSelf()方法。
- ☑ 提供 onBind()方法默认实现，其返回值是 null。
- ☑ 提供 onStartCommand()方法默认实现，它先发送 Intent 到工作队列，然后到 onHandleIntent()方法实现。

所有这些加在一起说明开发人员仅需要实现 onHandleIntent()方法来完成客户端提供的任务。由于 IntentService 类没有提供空参数的构造方法，因此需要提供一个构造方法。下面的代码是 IntentService 实现类的例子，在 onHandlerIntent()方法中，仅让线程休眠了 5 秒钟。

```
public class HelloIntentService extends IntentService {
    public HelloIntentService() {
        super("HelloIntentService");
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        long endTime = System.currentTimeMillis() + 5 * 1000;
        while (System.currentTimeMillis() < endTime) {
            synchronized (this) {
                try {
                    wait(endTime - System.currentTimeMillis());
                } catch (Exception e) {
                }
            }
        }
    }
}
```

这就是实现 IntentService 类所必须的全部操作：没有参数的构造方法和 onHandleIntent()方法。

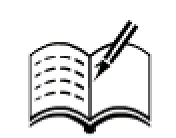
如果开发人员决定也重写其他回调方法，如 onCreate()、onStartCommand()或 onDestroy()，需要调用父类实现，这样 IntentService 能正确处理工作线程的生命周期。

例如，onStartCommand()方法必须返回默认实现。其代码如下：

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();
    return super.onStartCommand(intent,flags,startId);
}
```



除了 onHandleIntent()方法，仅有 onBind()方法不必调用父类实现，该方法在服务允许绑定时实现。

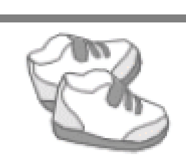


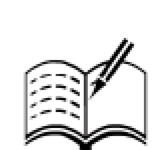
19.2.2　继承 Service 类

如上所述，使用 IntentService 类将简化启动服务的实现。然而，如果需要让服务处理多线程（取代使用工作队列处理启动请求），则可以继承 Service 类来处理各个 Intent。

作为对比，下面的例子通过实现 Service 类来完成与上面例子（实现 IntentService）完全相同的任务。对于每次启动请求，它使用工作线程来执行任务并每次处理一个请求。其代码如下：

```
public class HelloService extends Service {
    private Looper mServiceLooper;
    private ServiceHandler mServiceHandler;
    private final class ServiceHandler extends Handler {
        public ServiceHandler(Looper looper) {
            super(looper);
        }
        @Override
        public void handleMessage(Message msg) {
            long endTime = System.currentTimeMillis() + 5 * 1000;
            while (System.currentTimeMillis() < endTime) {
                synchronized (this) {
                    try {
                        wait(endTime - System.currentTimeMillis());
                    } catch (Exception e) {
                    }
                }
            }
            stopSelf(msg.arg1);
        }
    }
    @Override
    public void onCreate() {
        HandlerThread thread = new HandlerThread("ServiceStartArguments", Process.THREAD_PRIORITY_
BACKGROUND);
        thread.start();
        mServiceLooper = thread.getLooper();
        mServiceHandler = new ServiceHandler(mServiceLooper);
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Toast.makeText(this, "service starting", Toast.LENGTH_SHORT).show();
        Message msg = mServiceHandler.obtainMessage();
        msg.arg1 = startId;
        mServiceHandler.sendMessage(msg);
        return START_STICKY;
    }
```






```
        @Override
        public IBinder onBind(Intent intent) {
            return null;
        }
        @Override
        public void onDestroy() {
            Toast.makeText(this, "service done", Toast.LENGTH_SHORT).show();
        }
}
```

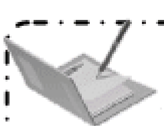
如上所示，这比使用 IntentService 麻烦了不少。

然而，由于开发人员自己处理 onStartCommand()方法调用，所以可以同时处理多个请求。这与示例代码不同，但是如果需要，就可以为每次请求创建一个新线程并且立即运行它们（避免等待前一个请求结束）。

onStartCommand()方法必须返回一个整数，该值用来描述系统停止服务后如何继续服务（如前所述，IntentService 默认实现已经处理了这些，开发人员也可以进行修改）。onStartCommand()方法返回值必须是表 19.1 所列常量值之一。

表 19.1　onStartCommand()方法返回值的常量值

常　量　值	说　　明
START_NOT_STICKY	如果系统在 onStartCommand()方法返回后停止服务，不重新创建服务，除非有 PendingIntent 要发送。在避免在不必要时运行服务和应用程序能简单地重启任何未完成工作时，这是最佳选择
START_STICKY	如果系统在 onStartCommand()方法返回后停止服务，重新创建服务并调用 onStartCommand()方法，但是不重新发送最后的 Intent。相反，系统使用空 Intent 调用 onStartCommand()方法，除非有 PendingIntent 来启动服务。此时，这些 Intent 会被发送。这适合多媒体播放器（或者类似服务），它们不执行命令但是无限期运行并等待工作
START_REDELIVER_INTENT	如果系统在 onStartCommand()方法返回后停止服务，重新创建服务并使用发送给服务的最后 Intent 调用 onStartCommand()方法。全部 PendingIntent 依次发送。这适合积极执行应该立即恢复工作的服务，如下载文件

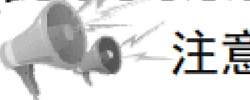


说明：

表 19.1 中的常量都定义在 Service 类中。

19.2.3　启动服务

开发人员可以从 Activity 或者其他应用程序组件通过传递 Intent 对象（指定要启动的服务）到 startService()方法启动服务。Android 系统调用服务的 onStartCommand()方法并将 Intent 传递给它。



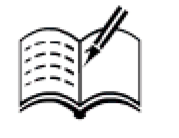
注意：

不要直接调用 onStartCommand()方法。



例如，Activity 能使用显式 Intent 和 startService()方法启动前面章节的示例服务（HelloService），其代码如下：

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```



startService()方法立即返回，然后 Android 系统调用服务的 onStartCommand()方法。如果服务还没有运行，系统首先调用 onCreate()方法，接着调用 onStartCommand()方法。

如果服务没有提供绑定，startService()方法发送的 Intent 是应用程序组件和服务之间唯一的通信模式。然而，如果开发人员需要服务返回结果，则启动该服务的客户端能为广播（使用 getBroadcast()方法）创建 PendingIntent 并通过启动服务的 Intent 发送它。服务接下来能使用广播来发送结果。

多个启动服务的请求导致服务的 onStartCommand()方法，然而仅需要一个停止方法（stopSelf()或 stopService()方法）来停止服务。

19.2.4 停止服务

启动服务必须管理自己的生命周期。即系统不会停止或销毁服务，除非它必须回收系统内存而且在 onStartCommand()方法返回后服务继续运行。因此，服务必须调用 stopSelf()方法停止自身，或者其他组件调用 stopService()方法停止服务。

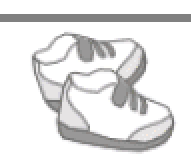
当使用 stopSelf()或 stopService()方法请求停止时，系统会尽快销毁服务。然而，如果服务同时处理多个 onStartCommand()方法调用请求，则处理完一个请求后，不应该停止服务。因为可能收到一个新的启动请求（在第一个请求结束后停止会终止第二个请求）。为了避免这个问题，开发人员可以使用 stopSelf(int)方法来确保停止服务的请求总是基于最近收到的启动请求。即当调用 stopSelf(int)方法时，同时将启动请求的 ID（发送给 onStartCommand()方法的 startId）传递给停止请求。这样如果服务在能够调用 stopSelf(int)方法前接收到新启动请求，ID 会不匹配因而服务不会停止。

注意：

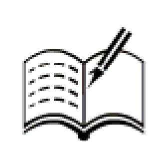
应用程序应该在任务完成后停止服务，以避免系统资源浪费和电池消耗。如果必要，其他组件能通过 stopService()方法停止服务。即便能够绑定服务，如果调用了 onStartCommand()方法就必须停止服务。

19.3 Bound Service 的使用

绑定服务是允许其他应用程序绑定并且与之交互的 Service 类实现类。为了提供绑定，开发人员必须实现 onBind()回调方法。该方法返回 iBinder 对象，它定义了客户端用来与服务交互的程序接口。



客户端能通过 bindService()方法绑定到服务。此时，客户端必须提供 ServiceConnection 接口的实现类，它监视客户端与服务之间的连接。bindService()方法立即返回，但是当 Android 系统创建客户端与服务之间的连接时，它调用 ServiceConnection 接口的 onServiceConnected()方法，来发送客户端用来与服务通信的 IBinder 对象。



Note

多个客户端能同时连接到服务。然而，仅当第一个客户端绑定时，系统调用服务的 onBind()方法来获取 IBinder 对象。系统接着发送同一个 IBinder 对象到其他绑定的客户端，但是不再调用 onBind()方法。

当最后的客户端与服务解绑定时，系统销毁服务（除非服务也使用 startService()方法启动）。

在实现绑定服务时，最重要的是定义 onBind()回调方法返回的接口，有 3 种方式可以定义这个接口。

☑ 继承 Binder 类

如果服务对应用程序私有并且与客户端运行于相同的进程（这非常常见），则应该继承 Binder 类来创建接口并且从 onBind()方法返回其一个实例。客户端接收 Binder 对象并使用它来直接访问 Binder 实现类或者 Service 类中可用公共方法。

当服务仅用于私有应用程序时，推荐使用该技术。只有当服务可以用于其他应用程序或者访问独立进程时，才不能使用该技术。

☑ 使用 Messenger

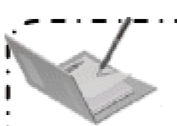
如果开发人员需要接口跨不同的进程工作，则可以使用 Messenger 来为服务创建接口。此时，服务定义 Handler 对象来响应不同类型的 Message 对象。Handler 是 Messenger 的基础，它能与客户端分享 IBinder，允许客户端使用 Message 对象向服务发送命令。此外，客户端能定义自己的 Messenger 对象，这样服务能发送回消息。

这是执行进程间通信（IPC）的最简单方式，因为 Messenger 类将所有请求队列化到单独的线程，这样开发人员就不必设计服务为线程安全。

☑ 使用 AIDL

AIDL（Android 接口定义语言）执行分解对象到原语的全部工作，以便操作系统能理解并且跨进程执行 IPC。使用 Messenger 创建接口，实际上将 AIDL 作为底层架构。如上所述，Messenger 在单个线程中将所有客户端请求队列化，这样服务每次收到一个请求。如果开发人员希望服务能同时处理多个请求，则可以直接使用 AIDL。此时，服务必须能处理多线程并且要保证线程安全。

为了直接使用 AIDL，开发人员必须创建定义编程接口的.aidl 文件。Android SDK 工具使用该文件来生成抽象类，它实现接口并处理 IPC，然后就可以在服务中使用。



说明：

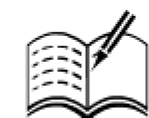
绝大多数应用程序不应该使用 AIDL 来创建绑定服务，因为它需要多线程能力而且会导致更加复杂的实现。因此，本章不进行讲解 AIDL 的使用。

19.3.1 继承 Binder 类

如果服务仅用于本地应用程序并且不必跨进程工作，则开发人员可以实现自己的 Binder 类来为客户端提供访问服务公共方法的方式。







Note

注意：

仅当客户端与服务位于同一个应用程序和进程时才有效，这也是最常见的情况。例如，音乐播放器需要绑定 Activity 到自己的服务来在后台播放音乐。

其实现步骤如下：

（1）在服务中，创建 Binder 类实例来完成下列操作之一。

☑　包含客户端能调用的公共方法。

☑　返回当前 Service 实例，其中包含客户端能调用的公共方法。

☑　返回服务管理的其他类的实例，其中包含客户端能调用的公共方法。

（2）从 onBind()回调方法中返回 Binder 类实例。

（3）在客户端，从 onServiceConnected()回调方法接收 Binder 类实例，并且使用提供的方法调用绑定服务。

说明：

服务和客户端必须位于同一个应用程序的原因是，客户端能转型返回对象并且适当地调用其方法。服务和客户端必须也位于同一个进程，因为该技术不支持跨进程。

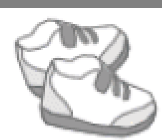
例如，下面的代码通过 Binder 实现类为客户端提供访问服务中方法的方法。

```
public class LocalService extends Service {
    private final IBinder binder = new LocalBinder();
    private final Random generator = new Random();
    public class LocalBinder extends Binder {
        LocalService getService() {
            return LocalService.this;
        }
    }
    @Override
    public IBinder onBind(Intent intent) {
        return binder;
    }
    public int getRandomNumber() {
        return generator.nextInt(100);
    }
}
```

LocalBinder 类为客户端提供了 getService()方法来获得当前 LocalService 的实例。这允许客户端调用服务中的公共方法。例如，客户端能从服务中调用 getRandomNumber()方法。

下面的 Activity 绑定到 LocalService，并且在单击按钮时调用 getRandomNumber()方法。

```
public class BindingActivity extends Activity {
    LocalService localService;
    boolean bound = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
```




Note

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, LocalService.class);
        bindService(intent, connection, Context.BIND_AUTO_CREATE);
    }
    @Override
    protected void onStop() {
        super.onStop();
        if (bound) {
            unbindService(connection);
            bound = false;
        }
    }
    public void onButtonClick(View v) {
        if (bound) {
            int num = localService.getRandomNumber();
            Toast.makeText(this, "获得随机数：" + num, Toast.LENGTH_SHORT).show();
        }
    }
    private ServiceConnection connection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            LocalBinder binder = (LocalBinder) service;
            localService = binder.getService();
            bound = true;
        }
        public void onServiceDisconnected(ComponentName arg0) {
            bound = false;
        }
    };
}
```

上面的代码演示客户端如何使用 ServiceConnection 实现类和 onServiceConnected()回调方法绑定到服务。

19.3.2 使用 Messenger 类

如果开发人员需要服务与远程进程通信，则可以使用 Messenger 来为服务提供接口，该技术允许不使用 AIDL 执行进程间通信（IPC）。

下面是关于如何使用 Messenger 的总结。

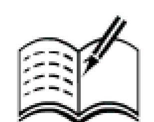
☑ 实现 Handler 的服务因为每次从客户端调用而收到回调。

☑ Handler 用于创建 Messenger 对象（它是 Handler 的引用）。

☑ Messenger 创建 IBinder，服务从 onBind()方法将其返回到到客户端。

☑ 客户端使用 IBinder 来实例化 Messenger，然后使用它来发送 Message 对象到服务。





☑　服务在其 Handler 的 handleMessage()方法接收 Message。

下面的例子演示了使用 Messenger 接口的服务。

```
public class MessengerService extends Service {
    static final int HELLO_WORLD = 1;
    class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
            case HELLO_WORLD:
                Toast.makeText(getApplicationContext(), "Hello World!", Toast.LENGTH_SHORT).show();
                break;
            default:
                super.handleMessage(msg);
            }
        }
    }
    final Messenger messenger = new Messenger(new IncomingHandler());
    @Override
    public IBinder onBind(Intent intent) {
        Toast.makeText(getApplicationContext(), "Binding", Toast.LENGTH_SHORT).show();
        return messenger.getBinder();
    }
}
```

Handler 中的 handleMessage()方法是服务接收 Message 对象的地方，并且根据 Message 类的 what 成员变量决定如何操作。

客户端需要完成的全部工作就是根据服务返回的 IBinder 创建 Messenger 并且使用 send()方法发送消息。例如，下面的 Activity 绑定到服务并发送 HELLO_WORLD 给服务。

```
public class ActivityMessenger extends Activity {
    Messenger messenger = null;
    boolean bound;
    private ServiceConnection connection = new ServiceConnection() {
        public void onServiceConnected(ComponentName className, IBinder service) {
            messenger = new Messenger(service);
            bound = true;
        }
        public void onServiceDisconnected(ComponentName className) {
            messenger = null;
            bound = false;
        }
    };
    public void sayHello(View v) {
        if (!bound)
            return;
        Message msg = Message.obtain(null, MessengerService.HELLO_WORLD, 0, 0);
        try {
            messenger.send(msg);
```

```
            } catch (RemoteException e) {
                e.printStackTrace();
            }
        }
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        }
        @Override
        protected void onStart() {
            super.onStart();
            bindService(new Intent(this, MessengerService.class), connection, Context.BIND_AUTO_CREATE);
        }
        @Override
        protected void onStop() {
            super.onStop();
            if (bound) {
                unbindService(connection);
                bound = false;
            }
        }
}
```

技巧：

本实例并没有演示服务如何响应客户端，如果开发人员希望服务响应，则需要在客户端也创建 Messenger。当客户端收到 onServiceConnected()回调方法时，它发送 Message 到服务。Message 的 replyTo 成员变量包含客户端的 Messenger。

19.3.3 绑定到服务

应用程序组件（客户端）能调用 bindService()方法绑定到服务。Android 系统接下来调用服务的 onBind()方法，它返回 IBinder 来与服务通信。

绑定是异步的。bindService()方法立即返回并且不返回 IBinder 到客户端。为了接收 IBinder，客户端必须创建 ServiceConnection 实例然后将其传递给 bindService()方法。ServiceConnection 包含系统调用发送 IBinder 的回调方法。

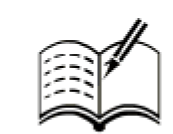
注意：

只有 Activity、Service 和 ContentProvider 能绑定到服务，BroadcastReceiver 不能绑定到服务。

如果需要从客户端绑定服务，需要完成以下操作。

（1）实现 ServiceConnection，这需要重写 onServiceConnected()和 onServiceDisconnected()两个回调方法。





（2）调用 bindService()方法，传递 ServiceConnection 实现。

（3）当系统调用 onServiceConnected()回调方法时，就可以使用接口定义的方法调用服务。

（4）调用 unbindService()方法解绑定。

当客户端销毁时，会将其从服务上解绑定。但是当与服务完成交互或者 Activity 暂停时，最好解绑定以便系统能及时停止不用的服务。

19.4　管理 Service 的生命周期

服务的生命周期比 Activity 简单很多，但却需要开发人员更加关注服务如何创建和销毁，因为服务在用户不知情时就可以在后台运行。服务的生命周期可以分成两个不同的路径。

☑　Started Service

当其他组件调用 startService()方法时，服务被创建。接着服务无限期运行，其自身必须调用 stopSelf()方法或者其他组件调用 stopService()方法来停止服务。当服务停止时，系统将其销毁。

☑　Bound Service

当其他组件调用 bindService()方法时，服务被创建。接着客户端通过 IBinder 接口与服务通信。客户端通过 unbindService()方法关闭连接。多个客户端能绑定到同一个服务并且当它们都解绑定时，系统销毁服务（服务不需要被停止）。

这两条路径并非完全独立，即开发人员可以绑定已经使用 startService()方法启动的服务，例如，后台音乐服务能使用包含音乐信息的 Intent 通过调用 startService()方法启动；然后，当用户需要控制播放器或者获得当前音乐信息时，可以调用 bindService()方法绑定 Activity 到服务，此时，stopService()和 stopSelf()方法直到全部客户端解绑定时才能停止服务。图 19.2 演示了两类服务的生命周期。

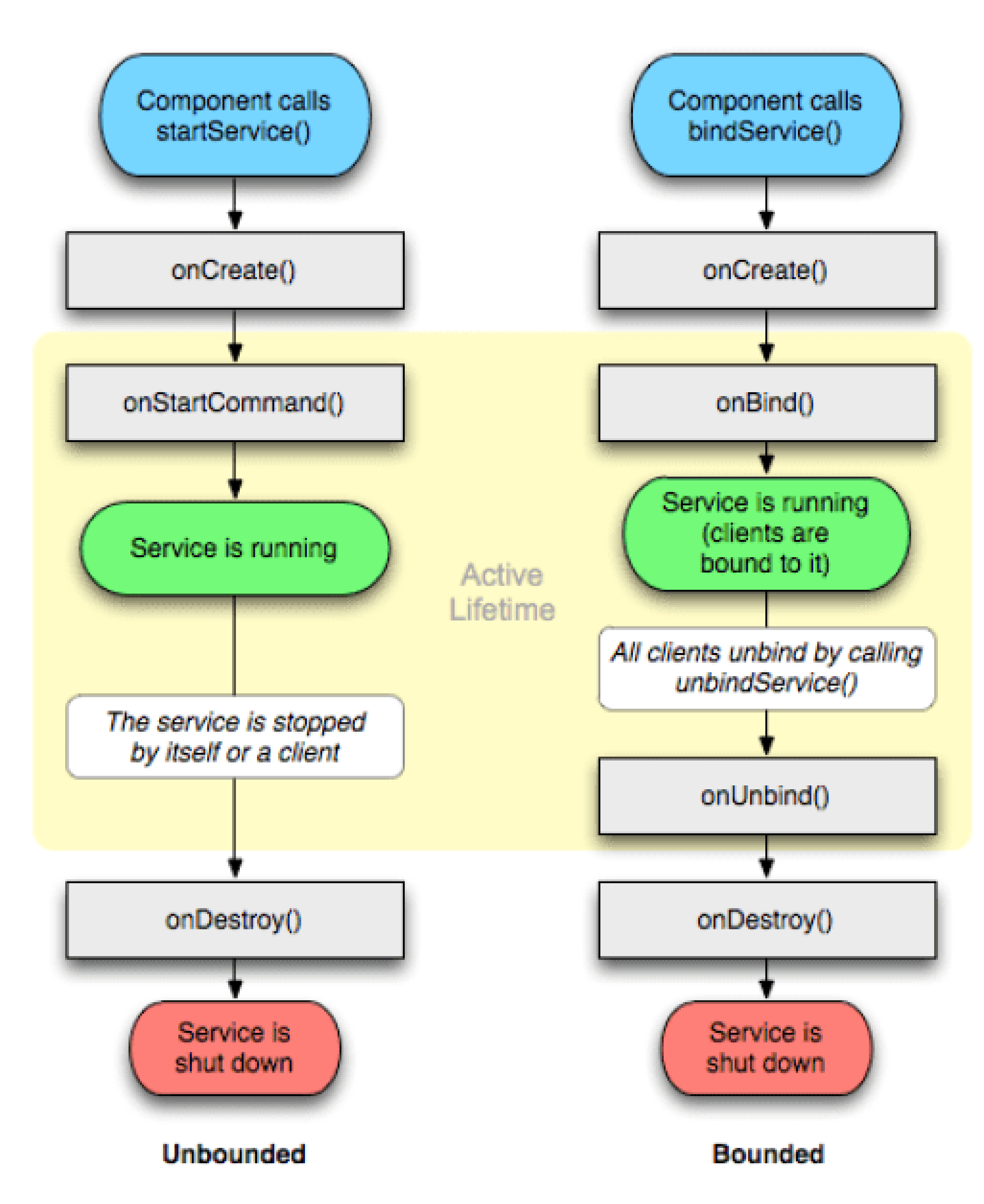
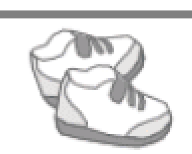


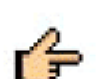
图 19.2　服务生命周期

19.5　综　合　应　用

19.5.1　继承 IntentService 输出当前时间

【例 19.1】 在 Eclipse 中创建 Android 项目，实现继承 IntentService 在后台输出当前时间。





Note

实例位置：光盘\MR\Instance\19\19.1

程序的开发步骤如下：

（1）修改 res\layout 包中的 main.xml 布局文件，设置背景图片并增加一个按钮，再设置按钮字体的内容、颜色和大小。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <Button
        android:id="@+id/current_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/current_time"
        android:textColor="@android:color/black"
        android:textSize="25dp" />
</LinearLayout>
```

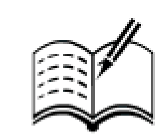
（2）创建 CurrentTimeService 类，它继承了 IntentService 类，用于在后台输出当前时间。其代码如下：

```
public class CurrentTimeService extends IntentService {
    public CurrentTimeService() {
        super("CurrentTimeService");                                    //调用父类非空构造方法
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        Time time = new Time();                                         //创建 Time 对象
        time.setToNow();                                                //设置时间为当前时间
        String currentTime = time.format("%Y-%m-%d %H:%M:%S");          //设置时间格式
        Log.i("CurrentTimeService", currentTime);                       //记录当前时间
    }
}
```

（3）创建 CurrentTimeActivity 类，它继承了 Activity 类。在 onCreate()方法中获得按钮控件并为其增加单击事件监听器。在监听器中，使用 Intent 启动服务。其代码如下：

```
public class CurrentTimeActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                                  //设置页面布局
        Button currentTime = (Button) findViewById(R.id.current_time);  //通过 id 值获得按钮
        currentTime.setOnClickListener(new View.OnClickListener() {     //增加单击事件监听器
            public void onClick(View v) {
                startService(new Intent(CurrentTimeActivity.this, CurrentTimeService.class));
```




Note

```
                                                //启动服务
            }
        });
    }
}
```

（4）修改 AndroidManifest.xml 文件，增加 Activity 和 Service 配置。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="18" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity android:name=".CurrentTimeActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service android:name=".CurrentTimeService"></service>
    </application>
</manifest>
```

运行本实例，界面如图 19.3 所示，单击“当前时间”按钮，会在 LogCat 中显示格式化了的当前时间，如图 19.4 所示。

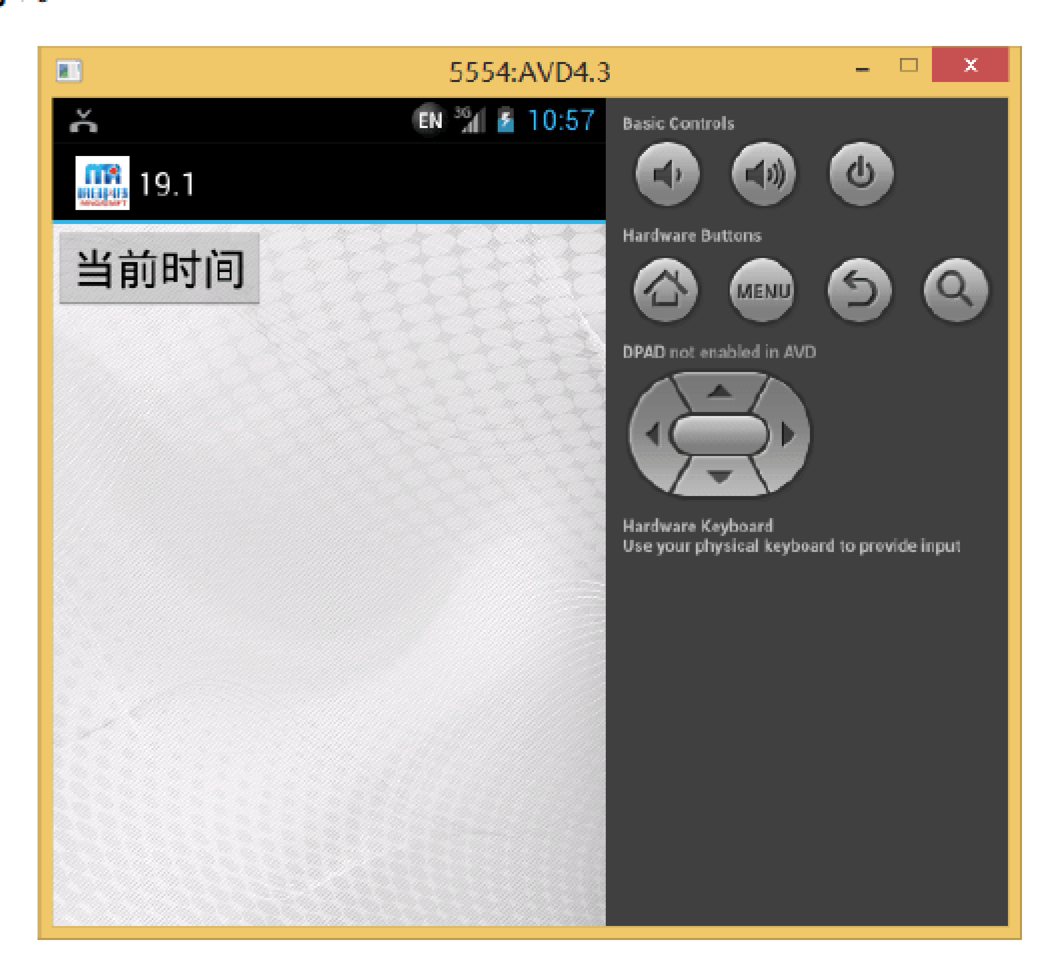
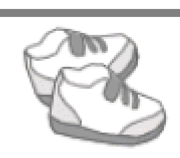


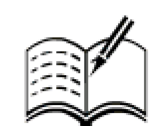
图 19.3　应用程序主界面

```
I    07-26 09:29:54.766    607    625    com.mingrisoft    CurrentTimeService    2012-07-26 09:29:54
```

图 19.4　LogCat 输出结果

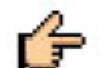


19.5.2　继承 Service 输出当前时间



Note

【例 19.2】 在 Eclipse 中创建 Android 项目，实现继承 Service 在后台输出当前时间。

 **实例位置：光盘\MR\Instance\19\19.2**

程序的开发步骤如下：

（1）修改 res\layout 包中的 main.xml 布局文件，设置背景图片并增加一个按钮，再设置按钮字体的内容、颜色和大小。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <Button
        android:id="@+id/current_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/current_time"
        android:textColor="@android:color/black"
        android:textSize="25dp" />
</LinearLayout>
```

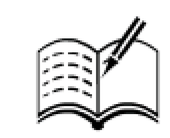
（2）创建 CurrentTimeService 类，它继承了 Service 类，并且重写了 onBind()和 onStartCommand()方法，其中 onStartCommand()方法用于在后台输出当前时间。其代码如下：

```
public class CurrentTimeService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Time time = new Time();                                      //创建 Time 对象
        time.setToNow();                                             //设置时间为当前时间
        String currentTime = time.format("%Y-%m-%d %H:%M:%S");       //设置时间格式
        Log.i("CurrentTimeService", currentTime);                    //记录当前时间
        return START_STICKY;
    }
}
```

（3）创建 CurrentTimeActivity 类，它继承了 Activity 类。在 onCreate()方法中获得按钮控件并为其增加单击事件监听器。在监听器中，使用 Intent 启动服务。其代码如下：

```
public class CurrentTimeActivity extends Activity {
    @Override
```

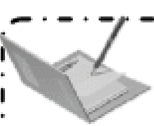





```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);                                    //设置页面布局
        Button currentTime = (Button) findViewById(R.id.current_time);   //通过 id 值获得按钮
        currentTime.setOnClickListener(new View.OnClickListener() {      //增加单击事件监听器
            public void onClick(View v) {
                startService(new Intent(CurrentTimeActivity.this, CurrentTimeService.class));
                                                                          //启动服务
            }
        });
    }
}
```

（4）修改 AndroidManifest.xml 文件，增加 Activity 和 Service 配置。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="18" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity android:name=".CurrentTimeActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service android:name=".CurrentTimeService"></service>
    </application>
</manifest>
```



说明：

本实例的运行效果与例 19.1 类似，详请参见图 19.3 和图 19.4。

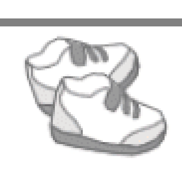
19.5.3　继承 Binder 类绑定服务显示时间

【例 19.3】 在 Eclipse 中创建 Android 项目，实现继承 Binder 类绑定服务，并显示当前时间。

实例位置：光盘\MR\Instance\19\19.3

程序的开发步骤如下：

（1）修改 res\layout 包中的 main.xml 布局文件，设置背景图片并增加一个按钮，再设置按钮字体的内容、颜色和大小。其代码如下：



Note

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <Button
        android:id="@+id/current_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/current_time"
        android:textColor="@android:color/black"
        android:textSize="25dp" />
</LinearLayout>
```

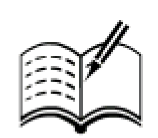
（2）创建 CurrentTimeService 类，它继承了 Service 类。内部类 LocalBinder 继承了 Binder 类，用于返回 CurrentTimeService 类的对象。getCurrentTime()方法用于返回当前时间。其代码如下：

```
public class CurrentTimeService extends Service {
    private final IBinder binder = new LocalBinder();
    public class LocalBinder extends Binder {
        CurrentTimeService getService() {
            return CurrentTimeService.this;                          //返回当前服务的实例
        }
    }
    @Override
    public IBinder onBind(Intent arg0) {
        return binder;
    }
    public String getCurrentTime() {
        Time time = new Time();                                      //创建 Time 对象
        time.setToNow();                                             //设置时间为当前时间
        String currentTime = time.format("%Y-%m-%d %H:%M:%S");       //设置时间格式
        return currentTime;
    }
}
```

（3）创建 CurrentTimeActivity 类，它继承了 Activity 类。在 onCreate()方法中设置布局。在 onStart()方法中，获得按钮控件并增加单击事件监听器。在监听器中，使用 bindService()方法绑定服务。在 onStop()方法中解除绑定。其代码如下：

```
public class CurrentTimeActivity extends Activity {
    CurrentTimeService cts;
    boolean bound;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
```

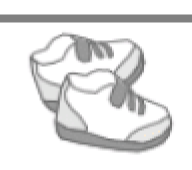





```
    protected void onStart() {
        super.onStart();
        Button button = (Button) findViewById(R.id.current_time);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(CurrentTimeActivity.this, CurrentTimeService.class);
                bindService(intent, sc, BIND_AUTO_CREATE);            //绑定服务
                if (bound) {                                          //如果绑定则显示当前时间
                    Toast.makeText(CurrentTimeActivity.this, cts.getCurrentTime(),
                                                        Toast.LENGTH_LONG).show();
                }
            }
        });
    }
    @Override
    protected void onStop() {
        super.onStop();
        if (bound) {
            bound = false;
            unbindService(sc);                                        //解绑定
        }
    }
    private ServiceConnection sc = new ServiceConnection() {
        public void onServiceDisconnected(ComponentName name) {
            bound = false;
        }
        public void onServiceConnected(ComponentName name, IBinder service) {
            LocalBinder binder = (LocalBinder) service;          //获得自定义的 LocalBinder 对象
            cts = binder.getService();                           //获得 CurrentTimeService 对象
            bound = true;
        }
    };
}
```

（4）修改 AndroidManifest.xml 文件，增加 Activity 和 Service 配置。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="18" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity android:name=".CurrentTimeActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```



```
        <service android:name=".CurrentTimeService" />
    </application>
</manifest>
```



Note

运行本实例，单击 Android 窗口中的“当前时间”按钮，会显示格式化了的当前时间，如图 19.5 所示。

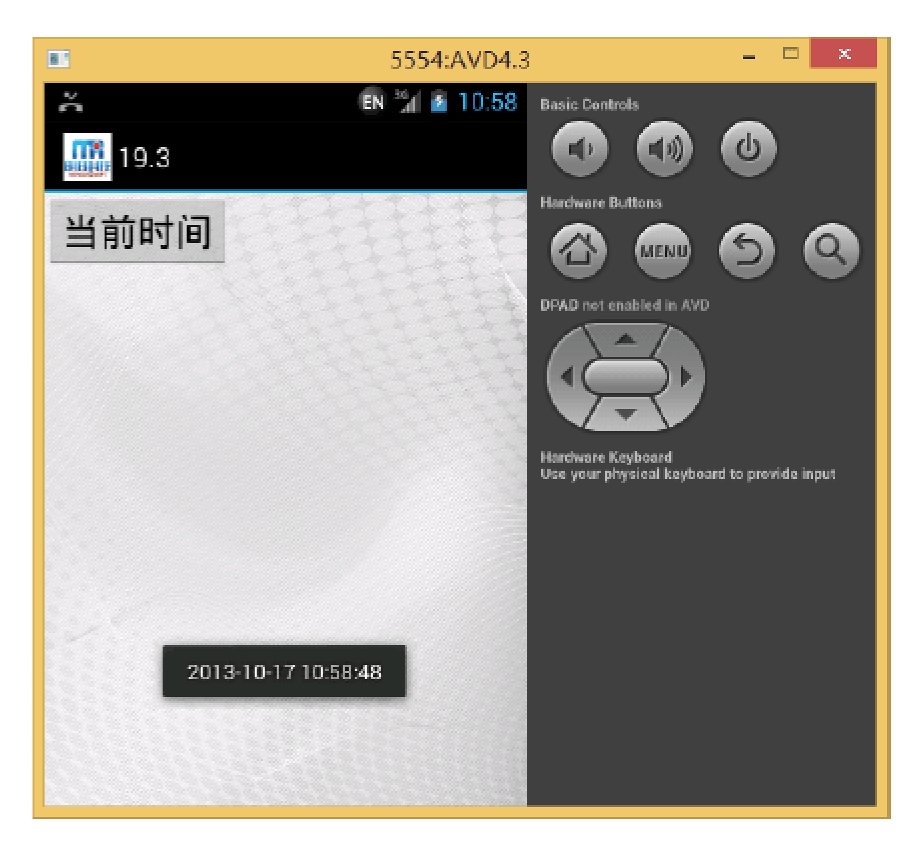


图 19.5　显示当前时间

19.5.4　使用 Messenger 类绑定服务显示时间

【例 19.4】 在 Eclipse 中创建 Android 项目，实现使用 Message 类绑定服务，并显示当前时间。

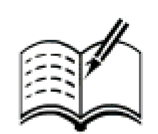
实例位置：光盘\MR\Instance\19\19.4

程序的开发步骤如下：

（1）修改 res\layout 包中的 main.xml 布局文件，设置背景图片并增加一个按钮，再设置按钮字体的内容、颜色和大小。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background"
    android:orientation="vertical" >
    <Button
        android:id="@+id/current_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/current_time"
        android:textColor="@android:color/black"
        android:textSize="25dp" />
</LinearLayout>
```



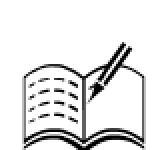


（2）创建 CurrentTimeService 类，它继承了 Service 类。内部类 IncomingHanlder 继承了 Handler 类，重写其 handleMessage()方法来显示当前时间。其代码如下：

```
public class CurrentTimeService extends Service {
    public static final int CURRENT_TIME = 0;
    private class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            if (msg.what == CURRENT_TIME) {
                Time time = new Time();                                    //创建 Time 对象
                time.setToNow();                                           //设置时间为当前时间
                String currentTime = time.format("%Y-%m-%d %H:%M:%S"); //设置时间格式
                Toast.makeText(CurrentTimeService.this, currentTime, Toast.LENGTH_LONG).show();
            } else {
                super.handleMessage(msg);
            }
        }
    }
    @Override
    public IBinder onBind(Intent intent) {
        Messenger messenger = new Messenger(new IncomingHandler());
        return messenger.getBinder();
    }
}
```

（3）创建 CurrentTimeActivity 类，它继承了 Activity 类。在 onCreate()方法中设置布局。在 onStart()方法中，获得按钮控件并增加单击事件监听器。在监听器中，使用 bindService()方法绑定服务。在 onStop()方法中解除绑定。其代码如下：

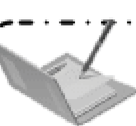
```
public class CurrentTimeActivity extends Activity {
    Messenger messenger;
    boolean bound;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    protected void onStart() {
        super.onStart();
        Button button = (Button) findViewById(R.id.current_time);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(CurrentTimeActivity.this, CurrentTimeService.class);
                bindService(intent, connection, BIND_AUTO_CREATE);       //绑定服务
                if (bound) {
                    Message message = Message.obtain(null, CurrentTimeService.CURRENT_TIME, 0, 0);
                    try {
                        messenger.send(message);
                    } catch (RemoteException e) {
```


Note

```
                        e.printStackTrace();
                    }
                }
            }
        });
    }
    @Override
    protected void onStop() {
        super.onStop();
        if (bound) {
            bound = false;
            unbindService(connection);                           //解绑定
        }
    }
    private ServiceConnection connection = new ServiceConnection() {
        public void onServiceDisconnected(ComponentName name) {
            messenger = null;
            bound = false;
        }
        public void onServiceConnected(ComponentName name, IBinder service) {
            messenger = new Messenger(service);
            bound = true;
        }
    };
}
```

（4）修改 AndroidManifest.xml 文件，增加 Activity 和 Service 配置。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mingrisoft"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="18" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity android:name=".CurrentTimeActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".CurrentTimeService" />
    </application>
</manifest>
```


说明：

本实例的运行效果与例 19.3 类似，详请参见图 19.5。





19.6 本章常见错误

开发 Android 程序时，如果在一个包中同时创建了 Service 和 Activity，那么它们是否处于同一进程呢？

一般来说，同一个包内的 Activity 和 Service，如果 Service 没有设置属性 android:process=".remote"，Service 和 Activity 就处于同一个进程中，由于一个进程只有一个 UI 线程，所以，Service 和 Activity 也是处在同一个线程里面；但是，如果为 Service 设置了属性 android:process=".remote"，那么，要在 Activity 中访问 Service 中的对象或者方法，就属于跨进程访问，这时，Service 和 Activity 就不在同一个进程中了。

19.7 本 章 小 结

本章详细介绍了 Android 四大组件之一的服务。对于服务而言，可以分成 Started 和 Bound 服务两大类。对于 Started 服务，有两种实现方式：继承 IntentService 类和继承 Service 类。对于 Bound 服务，有两种实现方式：继承 Binder 类和使用 Messenger 类。请读者认真区别各种方式，根据不同应用场合进行选择。

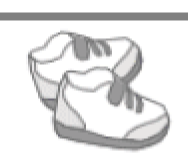
19.8 跟 我 上 机

参考答案：光盘\MR\跟我上机

在 Eclipse 中创建一个 Android 项目，当应用程序运行 1 分钟后，显示通知信息提醒用户保护视力。运行程序，在应用程序启动 1 分钟后会显示提示信息，单击打开后如图 19.6 所示。



图 19.6 视力保护程序



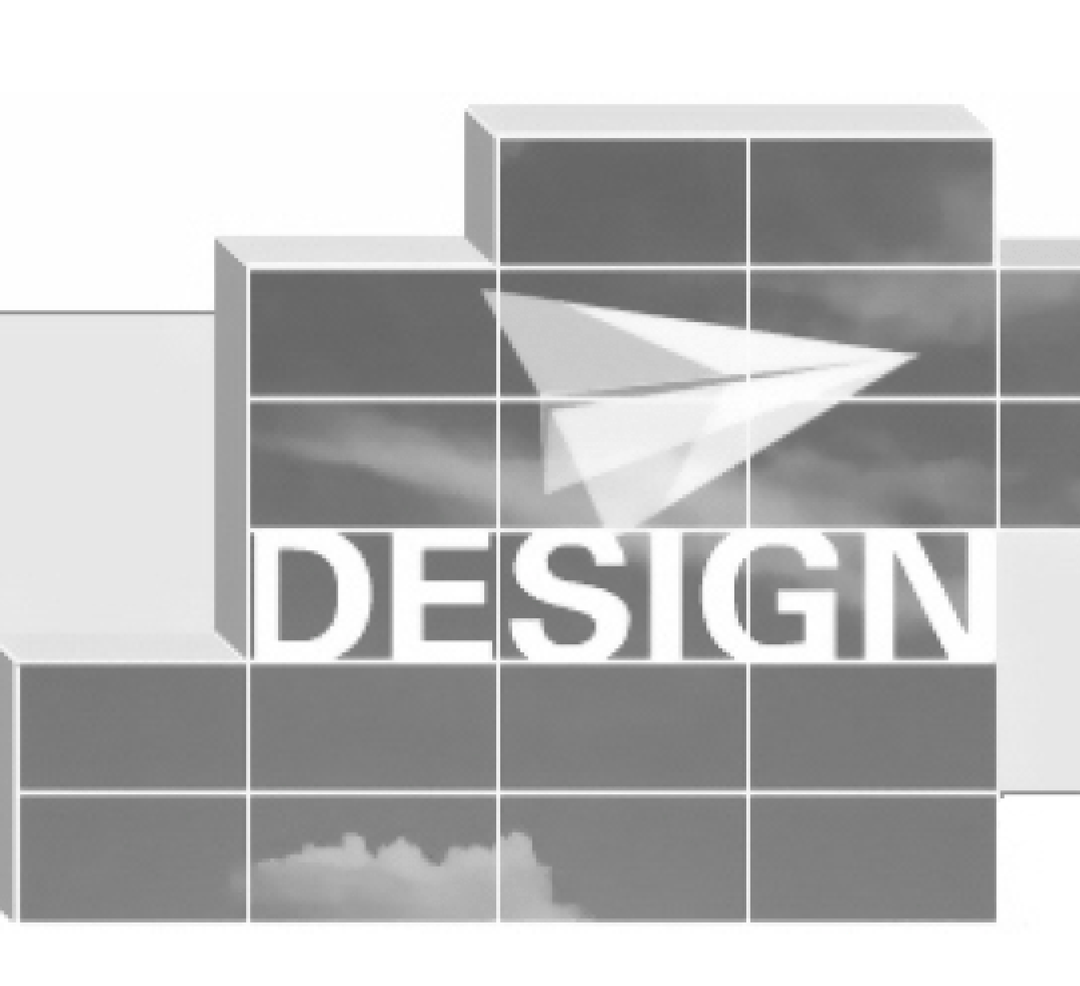


Note

本程序实现的关键是如何定义一个延时运行的服务，这里首先需要继承 Service 类，然后使用 Timer 类完成延时操作，并通过新建一个线程，控制在 60 秒后运行通知消息。其关键代码参考如下：

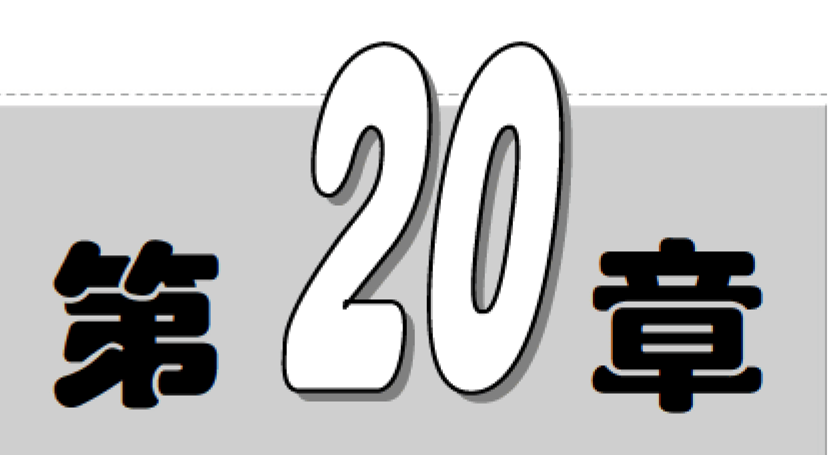
```
public class TimeService extends Service {
    private Timer timer;
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        super.onCreate();
        timer = new Timer(true);                                    //创建 Timer 对象
    }
    @Override
    public void onStart(Intent intent, int startId) {
        super.onStart(intent, startId);
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                String ns = Context.NOTIFICATION_SERVICE;
                //获得通知管理器
                NotificationManager manager = (NotificationManager) getSystemService(ns);
                Notification notification = new Notification(R.drawable.warning, getText(R.string.ticker_
text),System.currentTimeMillis());                                  //创建通知
                CharSequence contentTitle = getText(R.string.content_title);  //定义通知标题
                CharSequence contentText = getText(R.string.content_text);  //定义通知内容
                //创建 Intent 对象
                Intent intent = new Intent(TimeService.this, TimeActivity.class);
                PendingIntent contentIntent = PendingIntent.getActivity(TimeService.this, 0, intent,
Intent.FLAG_ACTIVITY_NEW_TASK);                             //创建 PendingIntent 对象
                //定义通知行为
                notification.setLatestEventInfo(TimeService.this, contentTitle, contentText, contentIntent);
                manager.notify(0, notification);                    //显示通知
                TimeService.this.stopSelf();                        //停止服务
            }
        }, 60000);
    }
}
```

第 3 篇



实战篇

- 第 20 章　Android 游戏——数独游戏
- 第 21 章　Android 应用——家庭理财通



Android 游戏——数独游戏

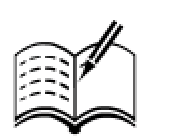
（视频讲解：28 分钟）

随着 Android 操作系统的普及，基于 Android 的应用需求越来越广，本章将使用最新的 Android 4.3 技术开发一个数独游戏。

本章能够完成的主要模块（已掌握的在方框中打勾）

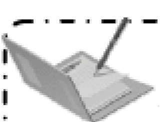
- ☐ 设计数独游戏的主窗体
- ☐ 设计虚拟键盘模块
- ☐ 设计游戏设置模块
- ☐ 设计关于模块
- ☐ 将数独游戏安装到 Android 手机上





20.1　需 求 分 析

数独游戏是一款比较传统的游戏，它由 81 个（9 行×9 列）单元格组成，玩家要试着在这些单元格中填入 1～9 的数字，使数字在每行、每列和每区（3 行×3 列的部分）中都只出现一次，游戏开始时，部分单元格中已经填入一些已知的数字，玩家只需要在剩下的空单元格中填入数字即可。



说明：

一道正确的数独谜题只有一个答案。

20.2　程序开发及运行环境

数独游戏的软件开发环境及运行环境具体如下。

☑ 操作系统：Windows 8。

☑ JDK 环境：Java SE Development KET(JDK) version 7。

☑ 开发工具：ADT Bundle（Eclipse+Android SDK+ADT）。

☑ 开发语言：Java、XML。

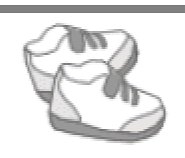
☑ 运行平台：Windows、Linux 各版本。

20.3　程序文件夹组织结构

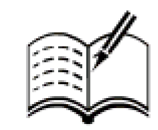
在编写项目代码之前，需要制定好项目的文件夹组织结构，如不同的 Java 包存放不同的窗体、公共类、数据模型、工具类或者图片资源等，这样不但可以保证团队开发的一致性，也可以规范系统的整体架构。创建完程序中可能用到的文件夹或者 Java 包之后，在开发时，只需将创建的类文件或者资源文件保存到相应的文件夹中即可。数独游戏的文件夹组织结构如图 20.1 所示。



图 20.1　文件夹组织结构



20.4 公共资源文件



Note

数独游戏中的公共资源文件主要有字符串资源文件、数组资源文件和颜色资源文件，设置完公共资源文件之后，在开发程序时，用户即可很方便地进行调用。本节将对数独游戏中的公共资源文件进行讲解。

20.4.1 字符串资源文件

字符串资源存储在 strings.xml 文件中，主要定义游戏中用到的公共字符串。其主要代码如下：

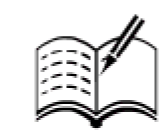
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Android 版的数独游戏</string>
    <string name="app_name">数独</string>
    <string name="btn1">继续</string>
    <string name="about_text">数独游戏是一款比较传统的游戏，它由 81 个（9 行*9 列）单元格组成，
玩家要试着在这些单元格中填入 1~9 的数字，使数字在每行、每列和每区（3 行*3 列的部分）中都只出现
一次，游戏开始时，部分单元格中已经填入一些已知的数字，玩家只需要在剩下的空单元格中填入数字。
    一道正确的数独谜题只有一个答案。
    </string>
    <string name="about_title">关于数独游戏</string>
    <string name="settings_label">设置...</string>
    <string name="settings_title">游戏设置</string>
    <string name="settings_shortcut">s</string>
    <string name="music_title">音乐</string>
    <string name="music_summary">播放背景音乐</string>
    <string name="hints_title">提示</string>
    <string name="hints_summary">是否显示提示</string>
    <!-- 开始游戏  -->
    <string name="new_game_title">难度</string>
    <string name="easy_label">简单</string>
    <string name="medium_label">一般</string>
    <string name="hard_label">高级</string>
    <string name="game_title">数独游戏</string>
    <string name="no_moves_label">不能填充任何数字</string>
    <string name="keypad_title">键盘</string>
</resources>
```

20.4.2 数组资源文件

数组资源存储在 arrarys.xml 文件中，主要定义数独游戏中的 3 种难易程度。其主要代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <array name="difficulty">
```





Note

```
        <item>@string/easy_label</item>
        <item>@string/medium_label</item>
        <item>@string/hard_label</item>
    </array>
</resources>
```

20.4.3　颜色资源文件

颜色资源存储在 colors.xml 文件中，主要定义游戏中用到的各种背景色，如主界面背景色、填充数字的单元格背景色和提醒背景色等。其主要代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <color name="background">#75FF6600</color>
        <color name="puzzle_background">#ffe6f0ff</color>
        <color name="puzzle_hilite">#FFFFFFFF</color>
        <color name="puzzle_light">#64c6d4ef</color>
        <color name="puzzle_dark">#6456648f</color>
        <color name="puzzle_foreground">#ff000000</color>
        <color name="puzzle_hint_0">#64ff0000</color>
        <color name="puzzle_hint_1">#6400ff80</color>
        <color name="puzzle_hint_2">#2000ff80</color>
        <color name="puzzle_selected">#64ff8000</color>
    </resources>
```

20.5　游戏主窗体设计

主窗体是程序操作过程中必不可少的，它是与用户交互中的重要环节。通过主窗体，用户可以调用系统相关的各子模块，快速掌握本系统中所实现的各个功能。数独游戏的主窗体主要为用户提供继续游戏、新建游戏、查看数据游戏规则及退出游戏的链接按钮。主窗体的运行效果如图 20.2 所示。

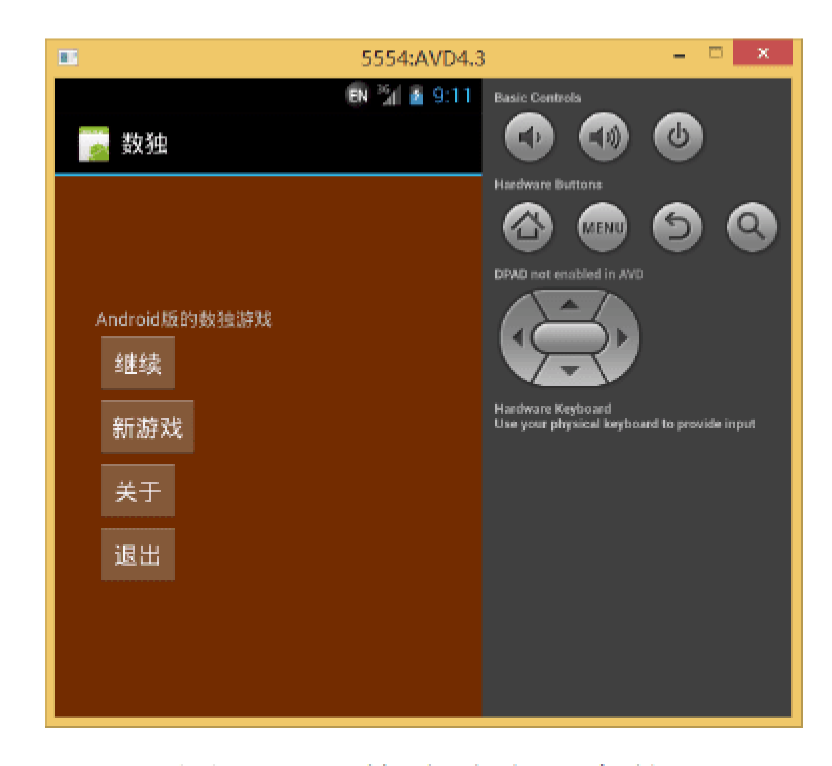
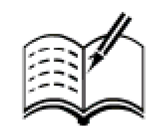


图 20.2　数独游戏主窗体



20.5.1 设计系统主窗体布局文件



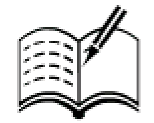
数独游戏的主窗体有两种布局方式：一种针对竖屏，一种针对横屏，其中，针对竖屏的布局文件存放在 res\layout 目录下。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:background="@color/background"
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="30dip"
        >
        <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="@string/hello"
            />
        <Button android:id="@+id/button1"
             android:text="@string/btn1"
             android:layout_height="wrap_content"
             android:layout_width="wrap_content"/>
        <Button android:id="@+id/button2"
             android:text="新游戏"
             android:layout_height="wrap_content"
             android:layout_width="wrap_content"/>
        <Button android:id="@+id/button3"
             android:text="关于"
             android:layout_height="wrap_content"
             android:layout_width="wrap_content"/>
        <Button android:id="@+id/button4"
             android:text="退出"
             android:layout_height="wrap_content"
             android:layout_width="wrap_content"/>
        </LinearLayout>
</LinearLayout>
```

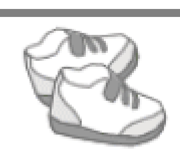
针对横屏的布局文件存放在 res\layout-land 目录下。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

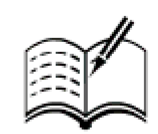





```
android:background="@color/background"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="15dip"
>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:paddingLeft="20dip"
    android:paddingRight="20dip"
         >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:layout_marginBottom="20dip"
    android:textSize="24.5sp"
    />
<TableLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:stretchColumns="*"
     >
     <TableRow>
         <Button android:id="@+id/button1"
              android:text="@string/btn1"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"/>
         <Button android:id="@+id/button2"
              android:text="新游戏"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"/>
         <Button android:id="@+id/button3"
              android:text="关于"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"/>
         <Button android:id="@+id/button4"
              android:text="退出"
              android:layout_height="wrap_content"
              android:layout_width="wrap_content"/>
     </TableRow>
</TableLayout>
</LinearLayout>
</LinearLayout>
```



20.5.2　为界面中的按钮添加监听事件

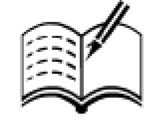


Note

在 com.wgh.sudoku 包中创建一个 SudokuActivity.java 文件，该文件主要是为界面中的按钮添加监听事件。其代码如下：

```
public class SudokuActivity extends Activity implements OnClickListener {
    private static final String TAG="Sudoku";
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        View continueButton=this.findViewById(R.id.button1);  //为“继续”按钮绑定单击事件
        continueButton.setOnClickListener(this);
        View newButton=this.findViewById(R.id.button2);
        newButton.setOnClickListener(this);
        View aboutButton=this.findViewById(R.id.button3);
        aboutButton.setOnClickListener(this);
        View exitButton=this.findViewById(R.id.button4);        //为“退出”按钮添加单击事件监听
        exitButton.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        //TODO Auto-generated method stub
        Intent i;
        switch (v.getId()){
            case R.id.button1:
                StartGame(GameActivity.DIFFICULTY_CONTINUE);
                break;
            case R.id.button2:
                openNewGameDialog();
                break;
            case R.id.button3:
                i=new Intent(this,About.class);
                startActivity(i);
                break;
            case R.id.button4:
                finish();
                break;
        }
    }
    @Override
    protected void onPause() {
        //TODO Auto-generated method stub
        super.onPause();
        Music.stop(this);
    }
    @Override
```



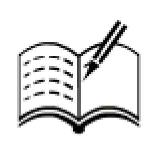



```
    protected void onResume() {
        super.onResume();
        Music.play(this,R.raw.jasmine);
    }
    private void openNewGameDialog() {
        new AlertDialog.Builder(this)
        .setTitle(R.string.new_game_title)
        .setItems(R.array.difficulty,new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int i) {
                //TODO Auto-generated method stub
                StartGame(i);
            }
        })
        .show();
    }
    private void StartGame(int i) {
        //TODO Auto-generated method stub
        Log.d(TAG,"clicked on "+i);
//      startActivity(new Intent(this,GraphicsActivity.class));
        Intent intent=new Intent(this,GameActivity.class);
        intent.putExtra(GameActivity.KEY_DIFFICULTY, i);
        startActivity(intent);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //TODO Auto-generated method stub
        super.onCreateOptionsMenu(menu);
        MenuInflater inflater=getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        //TODO Auto-generated method stub
        super.onOptionsItemSelected(item);
        switch (item.getItemId()){
        case R.id.settings:
            startActivity(new Intent(this,SettingsActivity.class));
            return true;
        }
        return false;
    }
}
```

20.5.3　绘制数独游戏界面

在 com.wgh.sudoku 包中创建一个 PuzzleView.java 文件，该文件主要是绘制数独游戏的界面。



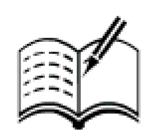


Note

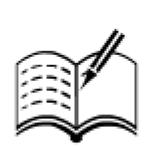
其代码如下：

```
public class PuzzleView extends View{
    private static final String TAG="sudoku";
    private final GameActivity game;
    private float width;
    private float height;
    private int selX;
    private int selY;
    private final Rect selRect=new Rect();
    //记录当前位置
    private static final String SELX="selX";
    private static final String SELY="selY";
    private static final String VIEW_STATE="viewState";
    private static final int ID=42;
    public PuzzleView(Context context) {
        super(context);
        this.game=(GameActivity)context;
        setFocusable(true);
        setFocusableInTouchMode(true);
        setId(ID);                                          //设置 ID 用于记录当前位置
    }
    //**************用于记录当前位置****************/
    @Override
    protected void onRestoreInstanceState(Parcelable state) {
        Log.d(TAG, "onRestoreInstanceState");
        Bundle bundle=(Bundle)state;
        select(bundle.getInt(SELX),bundle.getInt(SELY));
        super.onRestoreInstanceState(bundle.getParcelable(VIEW_STATE));
        return;
    }
    @Override
    protected Parcelable onSaveInstanceState() {
        Parcelable p=super.onSaveInstanceState();
        Log.d(TAG, "onSaveInstanceState");
        Bundle bundle=new Bundle();
        bundle.putInt(SELX, selX);
        bundle.putInt(SELY, selY);
        bundle.putParcelable(VIEW_STATE, p);
        return bundle;
    }
    /**************************************************/
    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        width=w/9f;
        height=h/9f;
        getRect(selX,selY,selRect);
        Log.d(TAG,"onSizeChanged:width"+width+"height"+height);
        //TODO Auto-generated method stub
        super.onSizeChanged(w, h, oldw, oldh);
```






```
}
@Override
protected void onDraw(Canvas canvas) {
        //TODO Auto-generated method stub
        Paint background=new Paint();
        background.setColor(getResources().getColor(R.color.puzzle_background));
        canvas.drawRect(0,0,getWidth(),getHeight(),background);
        //绘制网格线
        Paint dark=new Paint();
        dark.setColor(getResources().getColor(R.color.puzzle_dark));
        Paint hilite=new Paint();
        hilite.setColor(getResources().getColor(R.color.puzzle_hilite));
        Paint light=new Paint();
        light.setColor(getResources().getColor(R.color.puzzle_light));
        //绘制次要网格线
        for(int i=0;i<9;i++){
                canvas.drawLine(0, i*height, getWidth(), i*height, light);
                canvas.drawLine(0, i*height+1, getWidth(), i*height+1, hilite);
                canvas.drawLine(i*width, 0, i*width, getHeight(), light);
                canvas.drawLine(i*width+1, 0, i*width+1, getHeight(), hilite);
        }
        //绘制主要网格线
        for(int i=0;i<9;i++){
                if(i%3!=0){
                        continue;
                }else{
                        canvas.drawLine(0, i*height, getWidth(), i*height, dark);
                        canvas.drawLine(0, i*height+1, getWidth(), i*height+1, hilite);
                        canvas.drawLine(i*width, 0, i*width, getHeight(), dark);
                        canvas.drawLine(i*width+1, 0, i*width+1, getHeight(), hilite);
                }
        }
        //输出数字
        Paint foreground=new Paint(Paint.ANTI_ALIAS_FLAG);
        foreground.setColor(getResources().getColor(R.color.puzzle_foreground));
        foreground.setStyle(Style.FILL);
        foreground.setTextSize(height*0.75f);
        foreground.setTextScaleX(width/height);
        foreground.setTextAlign(Align.CENTER);                  //设置文字居中
        FontMetrics fm=foreground.getFontMetrics();
        float x=width/2;
        float y=height/2-(fm.ascent+fm.descent)/2;
        for(int i=0;i<9;i++){
                for(int j=0;j<9;j++){
                        canvas.drawText(this.game.getTileString(i,j), i*width+x, j*height+y, foreground);
                }
        }
        //绘制 hints
        if(SettingsActivity.getHints(getContext())){             //判断是否显示高亮提示
                Paint hint=new Paint();
```

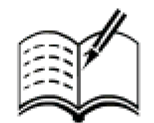

Note

```
                int c[]={getResources().getColor(R.color.puzzle_hint_0),
                        getResources().getColor(R.color.puzzle_hint_1),
                        getResources().getColor(R.color.puzzle_hint_2)};
                Rect r=new Rect();
                for(int i=0;i<9;i++){
                    for(int j=0;j<9;j++){
                        int mouseleft=9-game.getUsedTiles(i,j).length;
                        if(mouseleft<c.length){
                            getRect(i,j,r);
                            hint.setColor(c[mouseleft]);
                            canvas.drawRect(r,hint);

                        }
                    }
                }
            }
            //绘制选定区
            Log.d(TAG,"selRect"+selRect);
            Paint selected=new Paint();
            selected.setColor(getResources().getColor(R.color.puzzle_selected));
            canvas.drawRect(selRect,selected);
            super.onDraw(canvas);
        }
        @Override
        public boolean onKeyDown(int keyCode, KeyEvent event) {
            //TODO Auto-generated method stub
            Log.d(TAG,"onKeyDown:keycode="+keyCode+"event="+event);
            switch(keyCode){
                case KeyEvent.KEYCODE_DPAD_UP:
                    select(selX,selY-1);
                case KeyEvent.KEYCODE_DPAD_DOWN:
                    select(selX,selY+1);
                case KeyEvent.KEYCODE_DPAD_LEFT:
                    select(selX-1,selY);
                case KeyEvent.KEYCODE_DPAD_RIGHT:
                    select(selX+1,selY);
                    break;

                case KeyEvent.KEYCODE_0:
                case KeyEvent.KEYCODE_SPACE:
                    setSelectedTile(0);
                    break;
                case KeyEvent.KEYCODE_1:
                    setSelectedTile(1);
                    break;
                case KeyEvent.KEYCODE_2:
                    setSelectedTile(2);
                    break;
                case KeyEvent.KEYCODE_3:
                    setSelectedTile(3);
```






```
                break;
            case KeyEvent.KEYCODE_4:
                setSelectedTile(4);
                break;
            case KeyEvent.KEYCODE_5:
                setSelectedTile(5);
                break;
            case KeyEvent.KEYCODE_6:
                setSelectedTile(6);
                break;
            case KeyEvent.KEYCODE_7:
                setSelectedTile(7);
                break;
            case KeyEvent.KEYCODE_8:
                setSelectedTile(8);
                break;
            case KeyEvent.KEYCODE_9:
                setSelectedTile(9);
                break;
            case KeyEvent.KEYCODE_ENTER:
            case KeyEvent.KEYCODE_DPAD_CENTER:
                game.showKeyPadOrError(selX,selY);
                break;
            default:
                return super.onKeyDown(keyCode, event);
        }
        return true;
    }
    public void setSelectedTile(int tile) {
        if(game.setTileIfValid(selX,selY,tile)){
            invalidate();
        }else{
            Log.d(TAG,"setSelectedTile:invalid"+tile);
            startAnimation(AnimationUtils.loadAnimation(game, R.anim.shake));
        }
        //TODO Auto-generated method stub

    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        //TODO Auto-generated method stub
        if(event.getAction()!=MotionEvent.ACTION_DOWN){
            return super.onTouchEvent(event);
        }
        select((int)(event.getX()/width),(int)(event.getY()/height));
        game.showKeyPadOrError(selX,selY);
        Log.d(TAG,"onTouchEvent:x"+selX+",y"+selY);
        return true;
    }
    private void select(int x, int y) {
```

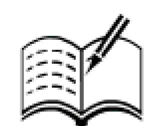
```
            invalidate(selRect);
            selX=Math.min(Math.max(x, 0), 8);
            selY=Math.min(Math.max(y, 0), 8);
            getRect(selX,selY,selRect);
            invalidate(selRect);
        }
        private void getRect(int x, int y, Rect rect) {
            rect.set((int)(x*width),(int)(y*height),(int)(x*width+width),(int)(y*height+height));
        }
}
```

20.5.4 数独游戏的实现算法

在 com.wgh.sudoku 包中创建一个 GameActivity.java 文件，该文件实现的功能主要有根据难易程度显示不同的游戏界面、保存并继续当前游戏和数独游戏的算法实现等。其代码如下：

```
public class GameActivity extends Activity {
        private static final String TAG = "sudoku";
        public static final String KEY_DIFFICULTY = "difficulty";
        public static final int DIFFICULTY_EASY = 0;
        public static final int DIFFICULTY_MEDIUM = 1;
        public static final int DIFFICULTY_HARD = 2;
        private int puzzle[] = new int[9 * 9];
        private PuzzleView puzzleView;
        private final int used[][][] = new int[9][9][];
        private final String easyPuzzle
                    = "360000000004230800000004200"
                    + "070460003820000014500013020"
                    + "001900000007048300000000045";
        private final String mediumPuzzle
                    = "650000070000506000014000005"
                    + "007009000002314700000700800"
                    + "500006300000201000030000097";
        private final String hardPuzzle
                    = "009000000080605020501078000"
                    + "000000700706040102004000000"
                    + "000720903090301080000000600";
        boolean success = false;                                    //判断是否成功
        //继续前一游戏
        private static final String PREF_PUZZLE="puzzle";
        protected static final int DIFFICULTY_CONTINUE=-1;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            //TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            Log.d(TAG, "onCreate");
            int diff = getIntent().getIntExtra(KEY_DIFFICULTY, DIFFICULTY_EASY);
            puzzle = getPuzzle(diff);                       //接收难度级别并返回一次数独游戏
```



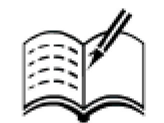



```
        Log.d(TAG, "onCreate11" + diff);
        calculateUsedTiles();                                //实现真正的游戏逻辑
        Log.d(TAG, "onCreate22" + diff);
        puzzleView = new PuzzleView(this);
        setContentView(puzzleView);
        puzzleView.requestFocus();
         getIntent().putExtra(KEY_DIFFICULTY, DIFFICULTY_CONTINUE); //恢复已保存的游戏
    }
    //获取游戏的难易程序
    private int[] getPuzzle(int diff) {
        String puz;
        switch (diff) {
        case DIFFICULTY_CONTINUE:
            puz=getPreferences(MODE_PRIVATE).getString(PREF_PUZZLE,easyPuzzle);
            break;
        case DIFFICULTY_HARD:
            puz = hardPuzzle;
            break;
        case DIFFICULTY_MEDIUM:
            puz = mediumPuzzle;
            break;
        case DIFFICULTY_EASY:
        default:
            puz = easyPuzzle;
            break;
        }
        return fromPuzzleString(puz);
    }
    public void showKeyPadOrError(int x, int y) {
        int tiles[] = getUsedTiles(x, y);
        if (tiles.length == 9) {
            Toast toast = Toast.makeText(this, R.string.no_moves_label,
                        Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.CENTER, 0, 0);
            toast.show();
        } else {
            Log.d(TAG, "showKeyPad:used=" + toPuzzleString(tiles));
            Dialog v = new KeyPad(this, tiles, puzzleView);
            v.show();
        }
    }
    private String toPuzzleString(int[] puz) {
        StringBuilder buf = new StringBuilder();
        for (int element : puz) {
            buf.append(element);
        }
        return buf.toString();
    }
    public boolean setTileIfValid(int x, int y, int value) {
        int tiles[] = getUsedTiles(x, y);
        if (value != 0) {
```




Note

```
                for (int tile : tiles) {
                    if (tile == value) {
                        return false;
                    }
                }
            }
            setTile(x, y, value);
            calculateUsedTiles();                                   //实现真正的游戏逻辑
            /**************** 判断游戏是否成功 *************************/
            success = true;
            label: for (int i = 0; i < 9; i++) {
                for (int j = 0; j < 9; j++) {
                    if (getTile(i, j) == 0) {
                        success = false;
                        break label;
                    }
                }
            }
            if (success) {
                Log.d(TAG, "数独游戏成功！");
                //弹出带“确定”按钮的提示对话框
                new AlertDialog.Builder(GameActivity.this)
                        .setTitle(TAG)
                        .setMessage("恭喜您，成功！")
                        .setPositiveButton("确定",
                                new DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
                                            int which) {
                                        finish();           //返回游戏主界面
                                    }
                                }).show();
            }
            /**********************************************************/
            return true;
        }
        private void calculateUsedTiles() {
            for (int i = 0; i < 9; i++) {
                for (int j = 0; j < 9; j++) {
                    used[i][j] = calculateUsedTiles(i, j);
                }
            }
        }
        private int[] calculateUsedTiles(int x, int y) {
            int c[] = new int[9];
            //水平方向
            for (int i = 0; i < 9; i++) {
                if (i == y) {
                    continue;
                }
                int t = getTile(x, i);
```

```
            if (t != 0) {
                c[t - 1] = t;
            }
        }
        //垂直方向
        for (int i = 0; i < 9; i++) {
            if (i == x) {
                continue;
            }
            int t = getTile(i, y);
            if (t != 0) {
                c[t - 1] = t;
            }
        }
        int startx = (x / 3) * 3;
        int starty = (y / 3) * 3;
        for (int i = startx; i < startx + 3; i++) {
            for (int j = starty; j < starty + 3; j++) {
                if (i == x && j == y) {
                    continue;
                }
                int t = getTile(i, j);
                if (t != 0) {
                    c[t - 1] = t;
                }
            }
        }
        int nused = 0;
        for (int t : c) {
            if (t != 0) {
                nused++;
            }
        }
        int cl[] = new int[nused];
        nused = 0;
        for (int t : c) {
            if (t != 0) {
                cl[nused++] = t;
            }
        }
        return cl;
    }
    /**
     * 功能：获取指定单元格中的数字
     * @param x
     * @param y
     * @return
     */
    private int getTile(int x, int y) {
        //TODO Auto-generated method stub
        return puzzle[y * 9 + x];
```

Note

```
    }
    /**
     * 功能：设置指定单元格中的数字
     * @param x
     * @param y
     * @param value
     */
    private void setTile(int x, int y, int value) {
        puzzle[y * 9 + x] = value;
    }
    protected int[] getUsedTiles(int x, int y) {
        return used[x][y];
    }
    public String getTileString(int x, int y) {
        int v = getTile(x, y);
        if (v == 0) {
            return "";
        } else {
            return String.valueOf(v);
        }
    }
    static protected int[] fromPuzzleString(String string) {
        int[] puz = new int[string.length()];
        for (int i = 0; i < puz.length; i++) {
            puz[i] = string.charAt(i) - '0';
        }
        return puz;
    }
    @Override
    protected void onPause() {                                          //暂停游戏
        super.onPause();
        Music.stop(this);
        getPreferences(MODE_PRIVATE).edit()
        .putString(PREF_PUZZLE, toPuzzleString(puzzle)).commit();      //保存游戏当前状态
    }
    @Override
    protected void onResume() {                                         //恢复游戏
        super.onResume();
        Music.play(this,R.raw.lhydd);
    }
}
```

20.6 虚拟键盘模块设计

用户在数独游戏的游戏界面中填写数字时，单击空白处，会出现一个虚拟键盘，以便提示用户可以填写哪些数字。虚拟键盘的运行效果如图 20.3 所示。



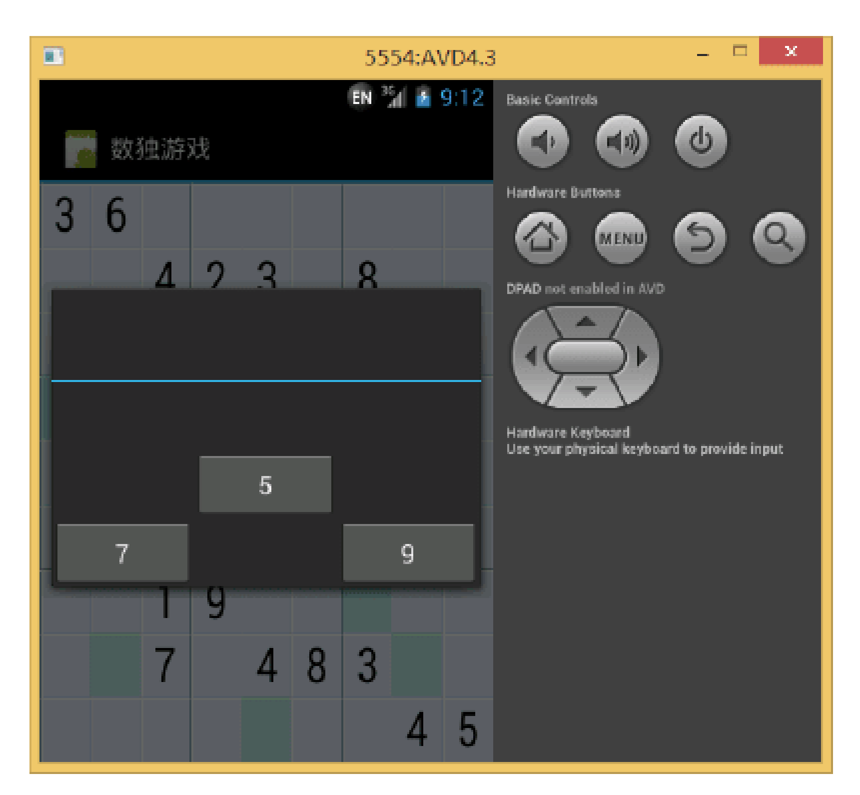



图 20.3　虚拟键盘

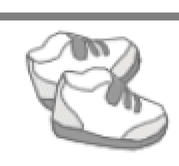
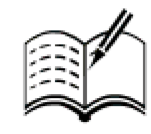
20.6.1　设计虚拟键盘布局文件

在 res\layout 目录下新建一个 keypad.xml 文件，用来作为虚拟键盘的布局文件，该布局文件使用 TableLayout 进行布局，并添加 9 个 Button 组件，分别表示 9 个数字按钮。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/keypad"
  android:orientation="vertical"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:stretchColumns="*">
  <TableRow>
     <Button android:id="@+id/keypad_1" android:text="1"/>
     <Button android:id="@+id/keypad_2" android:text="2"/>
     <Button android:id="@+id/keypad_3" android:text="3"/>
  </TableRow>
  <TableRow>
     <Button android:id="@+id/keypad_4" android:text="4"/>
     <Button android:id="@+id/keypad_5" android:text="5"/>
     <Button android:id="@+id/keypad_6" android:text="6"/>
  </TableRow>
    <TableRow>
     <Button android:id="@+id/keypad_7" android:text="7"/>
     <Button android:id="@+id/keypad_8" android:text="8"/>
     <Button android:id="@+id/keypad_9" android:text="9"/>
  </TableRow>
</TableLayout>
```

20.6.2　在虚拟键盘中显示可以输入的数字

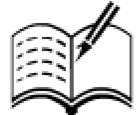
在 com.wgh.sudoku 包中创建一个 KeyPad.java 文件，该文件的布局文件设置为 keypad.xml。



Note

在 KeyPad.java 文件中，主要根据其他单元格的数字和数独游戏规则，在虚拟键盘中显示当前单元格可以输入的数字。其代码如下：

```
public class KeyPad extends Dialog{
	private static final String TAG="sudoku";
	private final View keys[]=new View[9];
	private View keypad;
	private final int useds[];
	private final PuzzleView puzzleView;
	public KeyPad(Context context,int useds[],PuzzleView puzzleView){
		super(context);
		this.useds=useds;
		this.puzzleView=puzzleView;
	}
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		//TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.keypad);
		findViews();
		for(int element:useds){
			keys[element-1].setVisibility(View.INVISIBLE);
		}
		setListeners();
	}
	@Override
	public boolean onKeyDown(int keyCode, KeyEvent event) {
		int tile=0;
		switch(keyCode){
		case KeyEvent.KEYCODE_0:
		case KeyEvent.KEYCODE_SPACE:tile=0;break;
		case KeyEvent.KEYCODE_1:tile=1;break;
		case KeyEvent.KEYCODE_2:tile=2;break;
		case KeyEvent.KEYCODE_3:tile=3;break;
		case KeyEvent.KEYCODE_4:tile=4;break;
		case KeyEvent.KEYCODE_5:tile=5;break;
		case KeyEvent.KEYCODE_6:tile=6;break;
		case KeyEvent.KEYCODE_7:tile=7;break;
		case KeyEvent.KEYCODE_8:tile=8;break;
		case KeyEvent.KEYCODE_9:tile=9;break;
		default:
			return super.onKeyDown(keyCode, event);
		}
		if(isValid(tile)){
			returnResult(tile);
		}
		return true;
	}
	/**
```






```
 *提取并保存软键盘的所有键和软键盘主窗口的视图
 */
	private void findViews() {
		keypad=findViewById(R.id.keypad);
		keys[0]=findViewById(R.id.keypad_1);
		keys[1]=findViewById(R.id.keypad_2);
		keys[2]=findViewById(R.id.keypad_3);
		keys[3]=findViewById(R.id.keypad_4);
		keys[4]=findViewById(R.id.keypad_5);
		keys[5]=findViewById(R.id.keypad_6);
		keys[6]=findViewById(R.id.keypad_7);
		keys[7]=findViewById(R.id.keypad_8);
		keys[8]=findViewById(R.id.keypad_9);
	}
	private void setListeners() {
		for(int i=0;i<keys.length;i++){
			final int t=i+1;
			keys[i].setOnClickListener(new View.OnClickListener() {
				@Override
				public void onClick(View v) {
					returnResult(0);

				}
			});
		}
	}
	private boolean isValid(int tile){
		for(int t:useds){
			if(tile==t){
				return false;
			}
		}
		return true;
	}
	private void returnResult(int tile) {
		puzzleView.setSelectedTile(tile);
		dismiss();
	}
}
```

20.7　游戏设置模块设计

游戏设置模块主要对背景音乐和是否显示提示进行设置，该模块中主要通过两个复选框实现。游戏设置模块的运行效果如图 20.4 所示。




Note

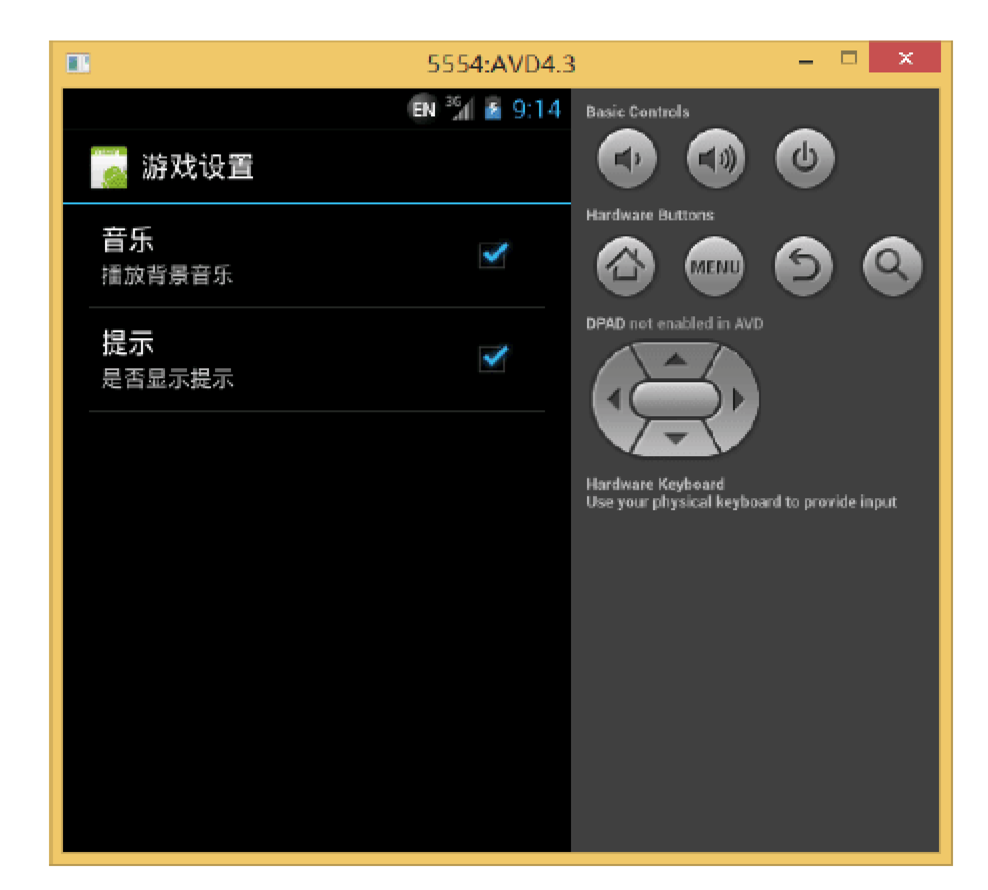


图 20.4　游戏设置

20.7.1　设计游戏设置布局文件

在 res\xml 目录下新建一个 settings.xml 文件，用来作为游戏设置窗体的布局文件，该布局文件主要使用两个 CheckBoxPreference 组件，用来作为复选框。其实现代码如下：

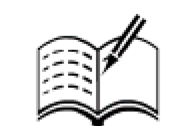
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
          android:key="music"
          android:title="@string/music_title"
          android:summary="@string/music_summary"
          android:defaultValue="true"/>
    <CheckBoxPreference
          android:key="hints"
          android:title="@string/hints_title"
          android:summary="@string/hints_summary"
          android:defaultValue="true"/>
</PreferenceScreen>
```

20.7.2　设置是否播放背景音乐和显示提示

在 com.wgh.sudoku 包中创建一个 SettingsActivity.java 文件，该文件的布局文件设置为 settings.xml。在 SettingsActivity.java 文件中，主要定义了两个方法，分别设置是否播放背景音乐和是否显示提示。其代码如下：

```
public class SettingsActivity extends PreferenceActivity {
      private static final String OPT_MUSIC="music";
      private static final boolean OPT_MUSIC_DEF=true;
```






```
        private static final String OPT_HINTS="hints";
        private static final boolean OPT_HINTS_DEF=true;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                //TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                addPreferencesFromResource(R.xml.settings);
        }
        public static boolean getMusic(Context context){
                return PreferenceManager.getDefaultSharedPreferences(context)
                .getBoolean(OPT_MUSIC,OPT_MUSIC_DEF);
        }
        public static boolean getHints(Context context){
                return PreferenceManager.getDefaultSharedPreferences(context)
                .getBoolean(OPT_HINTS,OPT_HINTS_DEF);
        }
}
```

20.7.3 控制背景音乐的播放与停止

在 com.wgh.sudoku 包中创建一个 Music.java 文件，该文件主要控制背景音乐的播放与停止。其代码如下：

```
public class Music {
        private static MediaPlayer mp = null;
        public static void play(Context context, int resource) {
                stop(context);
                if (SettingsActivity.getMusic(context)) {          //判断是否播放背景音乐
                        mp = MediaPlayer.create(context, resource);
                        mp.setLooping(true);                       //是否循环播放
                        mp.start();                                //开始播放
                }
        }
        public static void stop(Context context) {
                if (mp != null) {
                        mp.stop();
                        mp.release();
                        mp = null;
                }
        }
}
```

20.8 关于模块设计

关于模块主要显示数独游戏的相关规则，关于窗体的运行效果如图 20.5 所示。




Note



图 20.5　关于模块

20.8.1　设计关于窗体布局文件

在 res\layout 目录下新建一个 about.xml 文件，用来作为关于窗体的布局文件，该布局文件主要使用一个 TextView 组件实现数独游戏的相关规则。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="10dip"
  >
   <TextView
            android:id="@+id/about_content"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/about_text"
      />
</ScrollView>
```

20.8.2　显示关于信息

在 com.wgh.sudoku 包中创建一个 About.java 文件，该文件加载 about.xml 布局文件，以便显示关于数独游戏的规则信息。其代码如下：

```
public class About extends Activity {
     @Override
```



```
        protected void onCreate(Bundle savedInstanceState) {
            //TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.about);
        }
}
```



Note

20.9　将程序安装到 Android 手机上

Android 程序开发完成之后，需要安装到载有 Android 操作系统的手机上，那么如何将数独游戏安装到 Android 手机上呢？本节将详细介绍。

使用 adb 命令将数独游戏安装到 Android 模拟器上的步骤如下：

（1）开发完数独游戏后，在 Eclipse 中运行该程序，会在项目文件夹的 bin 文件夹下自动生成一个 apk 文件，如图 20.6 所示，将该 apk 文件复制到 Android SDK 安装路径下的 platform-tools 文件夹中。

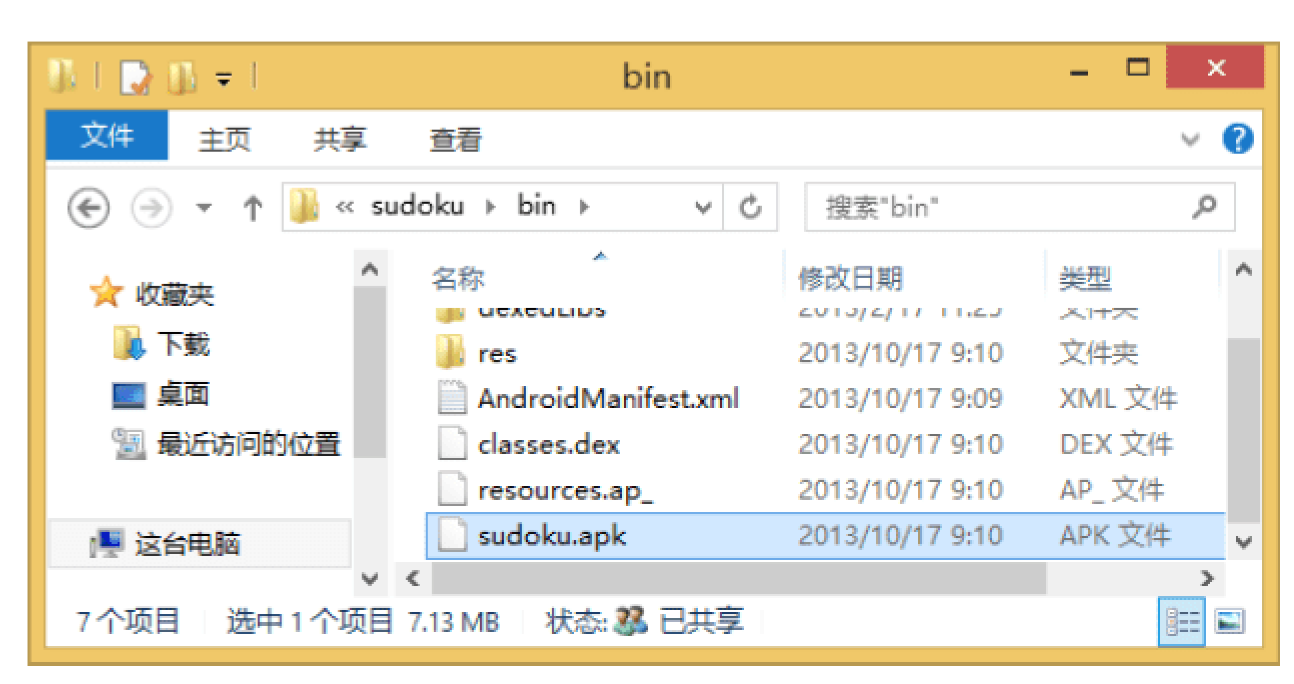


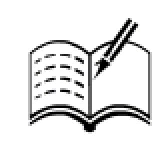
图 20.6　项目 bin 文件夹下自动生成的 apk 文件

（2）打开 cmd 命令提示窗口，首先把路径切换到 Android SDK 安装路径的 platform-tools 文件夹，然后使用 adb install 命令将 sudoku.apk 文件安装到 Android 模拟器上；如果要将 apk 文件安装到 Android 模拟器的 SD 卡上，则使用 adb install –s 命令，如图 20.7 所示。



图 20.7　使用 adb 命令安装数独游戏





Note

（3）安装完成后，显示 Success 成功信息，打开 Android 模拟器，可以看到安装的数独游戏，如图 20.8 所示。



图 20.8　安装的数独游戏

20.10　本 章 小 结

本章重点讲解了数独游戏的实现过程及安装过程。通过对本章的学习，读者应该能够熟悉 Android 应用的开发流程，并重点掌握数独游戏的游戏规则和实现算法。

第21章

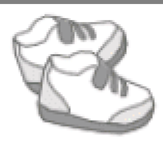
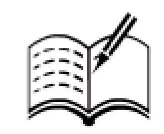
Android 应用——家庭理财通

（视频讲解：46 分钟）

随着 3G 智能手机的迅速普及，移动互联网时代已经离我们越来越近，作为互联网巨头 Google 推出的免费手机平台 Android，已经得到众多厂商和开发者的拥护，而随着 Android 手机操作系统的大热，基于 Android 的软件也越来越受到广大用户的欢迎。本章将使用 Android 4.3 技术开发一个家庭理财通系统，通过该系统，可以随时随地记录用户的收入及支出等信息。

本章能够完成的主要模块（已掌握的在方框中打勾）

- ☐ 使用 SQLite 数据库
- ☐ 创建公共类
- ☐ 设计登录模块
- ☐ 设计系统主窗体
- ☐ 设计收入管理模块
- ☐ 设计便签管理模块
- ☐ 设计系统设置模块

Note

21.1 需求分析

你是月光族吗？你能说出每月的钱都用到什么地方了吗？为了更好地记录您每月的收入及支出，这里开发了一款基于 Android 系统的家庭理财通软件。通过该软件，用户可以随时随地记录自己的收入、支出等信息。另外，为了保护自己的隐私，还可以为家庭理财通设置密码。

21.2 系统设计

21.2.1 系统目标

根据个人对家庭理财通软件的要求，制定目标如下。

☑ 操作简单方便、界面简洁美观。
☑ 方便对收入及支出进行增、删、改、查等操作。
☑ 通过便签方便地记录用户的计划。
☑ 能够通过设置密码保证程序的安全性。
☑ 系统运行稳定、安全可靠。

21.2.2 系统功能结构

家庭理财通的功能结构如图 21.1 所示。

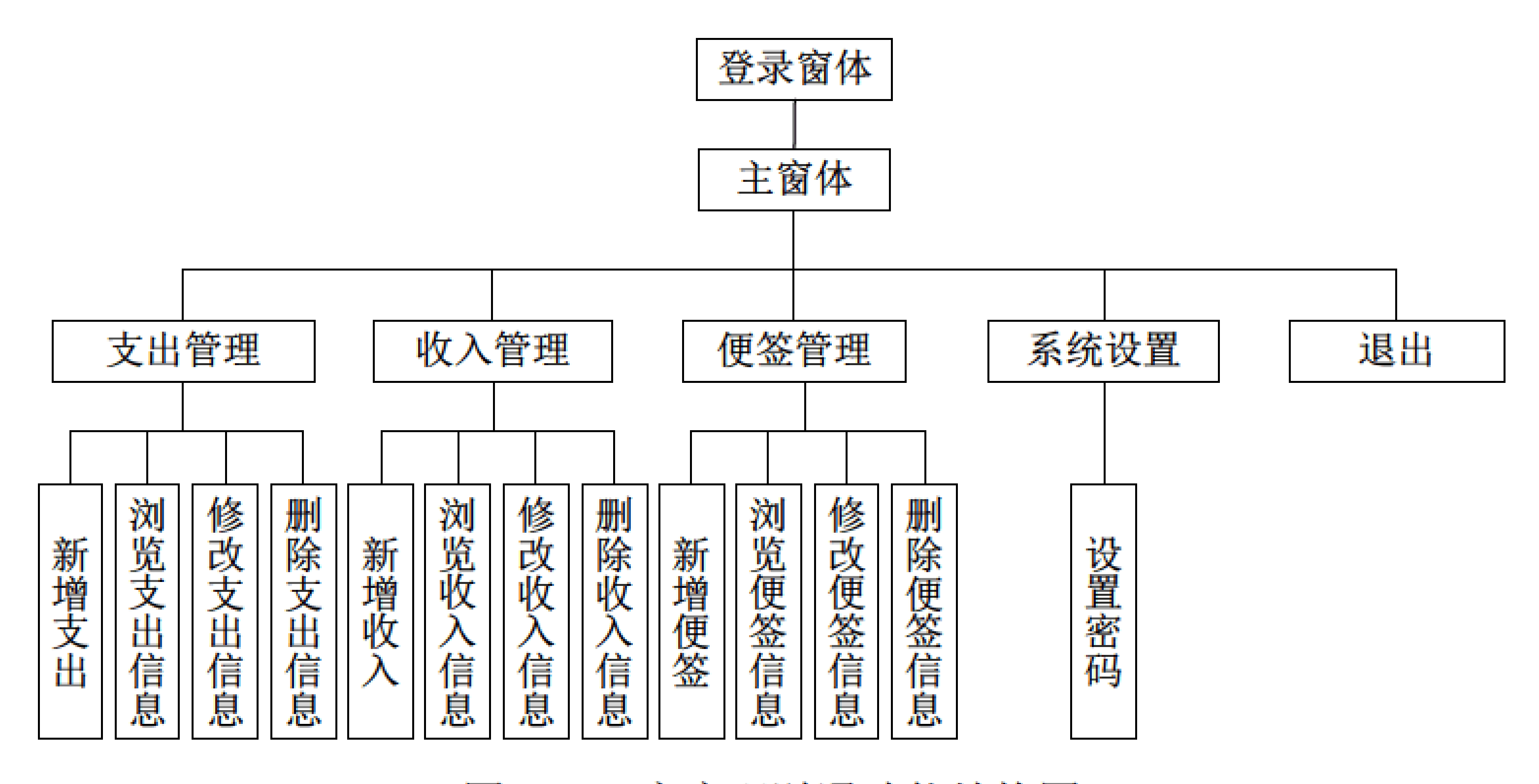
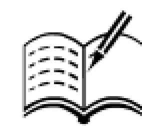


图 21.1　家庭理财通功能结构图

21.2.3 系统业务流程图

家庭理财通的业务流程图如图 21.2 所示。





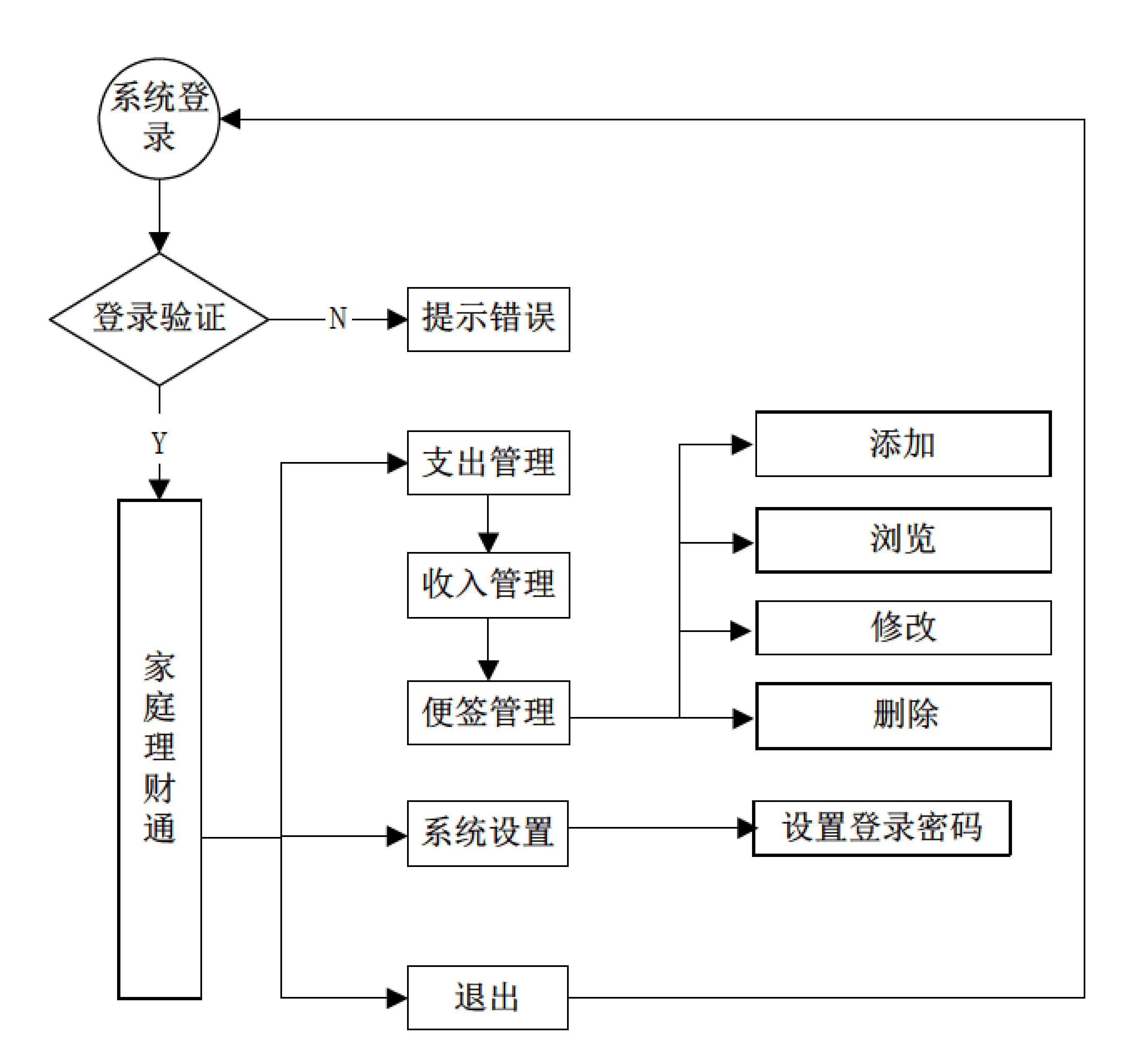


图 21.2　家庭理财通业务流程图

21.2.4　系统编码规范

开发应用程序常常需要以团队合作来完成，每个人负责不同的业务模块，为了使程序的结构与代码风格统一标准化，增加代码的可读性，需要在编码之前制定一套统一的编码规范。下面介绍家庭理财通系统开发中的编码规范。

1．数据库命名规范

☑　数据库

数据库以数据库相关英文单词或缩写进行命名，如表 21.1 所示。

表 21.1　数据库命名

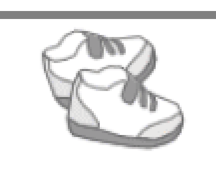
数据库名称	描　　述
account.db	家庭理财通数据库

☑　数据表

数据表以字母 tb 开头（小写），后面加数据表相关英文单词或缩写。下面将举例进行说明，如表 21.2 所示。

表 21.2　数据表命名

数据表名称	描　　述
tb_outaccount	支出信息表





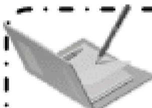
Note

☑ 字段

字段一律采用英文单词或词组（可利用翻译软件）命名，如找不到专业的英文单词或词组，可以用相同意义的英文单词或词组代替。下面将举例进行说明，如表 21.3 所示。

表 21.3 字段命名

字段名称	描 述
_id	编号
money	金额



说明：

在数据库中使用命名规范，有助于其他用户更好地理解数据表，及其表中各字段的内容。

2. 程序代码命名规范

（1）数据类型简写规则

程序中定义常量、变量或方法等内容时，常常需要指定类型。下面介绍一种常见的数据类型简写规则，如表 21.4 所示。

表 21.4 数据类型简写规则

数据类型	简 写	数据类型	简 写
整型	int	单精度浮点型	flt
字符串	str	双精度浮点型	dbl
布尔型	bl		

（2）组件命名规则

所有组件对象的名称都为自然名称的拼音简写，出现冲突可采用不同的简写规则。组件命名规则如表 21.5 所示。

表 21.5 组件命名规则

控 件	缩写形式	控 件	缩写形式
EditText	txt	ListView	lv
Button	btn	……	……
Spinner	sp		

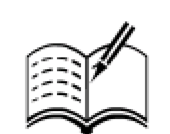
21.3 系统开发及运行环境

本系统的软件开发环境及运行环境具体如下。

☑ 操作系统：Windows 8。

☑ JDK 环境：Java SE Development KET(JDK) version 7。





☑ 开发工具：ADT Bundle（Eclipse+Android SDK+ADT）。
☑ 开发语言：Java、XML。
☑ 数据库管理软件：SQLite 3。
☑ 运行平台：Windows、Linux 各版本。

21.4　数据库与数据表设计

开发应用程序时，对数据库的操作是必不可少的，数据库设计是根据程序的需求及其实现功能所制定的，数据库设计的合理性将直接影响到程序的开发过程。

21.4.1　数据库分析

家庭理财通是一款运行在 Android 系统上的程序，在 Android 系统中，集成了一种轻量型的数据库，即 SQLite，该数据库是使用 C 语言编写的开源嵌入式数据库，支持的数据库大小为 2TB，使用该数据库，用户可以像使用 SQL Server 或 Oracle 数据库那样来存储、管理和维护数据。本系统采用了 SQLite 数据库，并且命名为 account.db，该数据库中用到了 4 个数据表，分别是 tb_flag、tb_inaccount、tb_outaccount 和 tb_pwd，如图 21.3 所示。

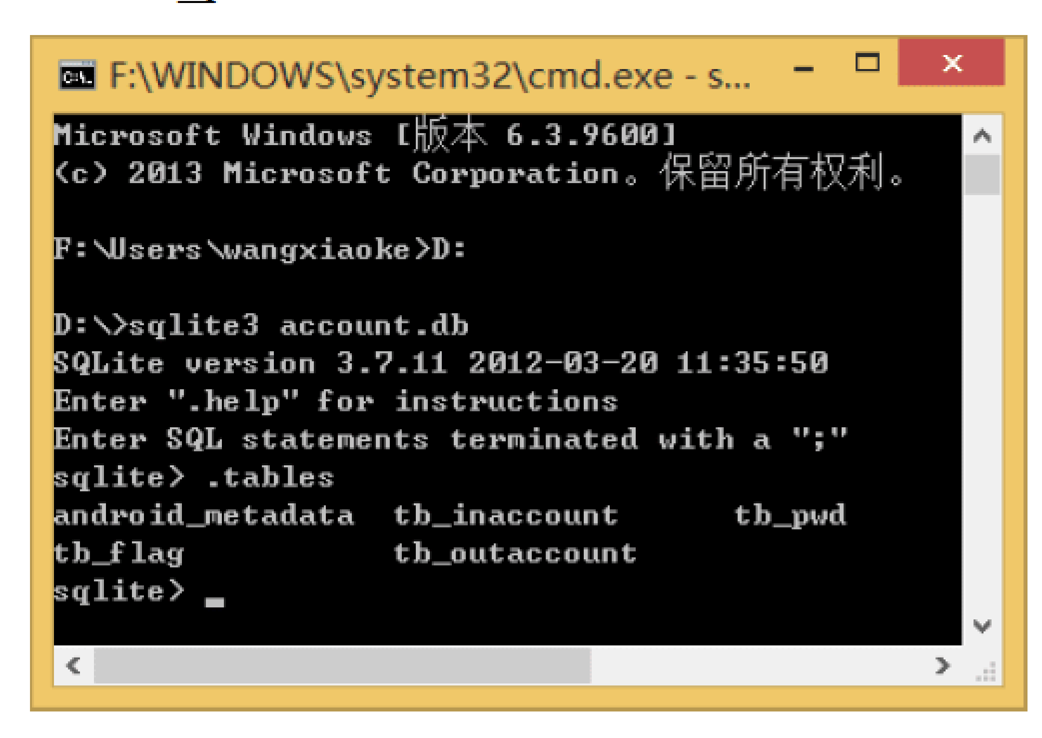
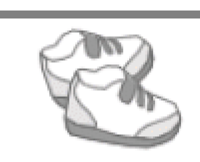


图 21.3　家庭理财通系统中用到的数据表

21.4.2　创建数据库

开发家庭理财通系统在创建数据库时，通过使用 SQLiteOpenHelper 类的构造函数来实现。其实现代码如下：

```
private static final int VERSION = 1;                          //定义数据库版本号
private static final String DBNAME = "account.db";             //定义数据库名
public DBOpenHelper(Context context)                           //定义构造函数
{
```



```
    super(context, DBNAME, null, VERSION);                    //重写基类的构造函数，以创建数据库
}
```

Note

技巧：

创建数据库时，也可以在 cmd 命令窗口中使用 sqlite3 命令打开 SQLite 数据库，然后使用 create database 语句创建，但这里需要注意的是，在 cmd 命令窗口中操作 SQLite 数据库时，SQL 语句最后需要加分号（;）。

21.4.3 创建数据表

在创建数据表前，首先要根据项目实际要求规划相关的数据表结构，然后在数据库中创建相应的数据表。

☑ tb_pwd（密码信息表）

tb_pwd 表用于保存家庭理财通的密码信息，该表的结构如表 21.6 所示。

表 21.6 密码信息表

字 段 名	数 据 类 型	主 键 否	描 述
password	varchar(20)	否	用户密码

☑ tb_outaccount（支出信息表）

tb_outaccount 表用于保存用户的支出信息，该表的结构如表 21.7 所示。

表 21.7 支出信息表

字 段 名	数 据 类 型	主 键 否	描 述
_id	integer	是	编号
money	decimal	否	支出金额
time	varchar(10)	否	支出时间
type	varchar(10)	否	支出类别
address	varchar(100)	否	支出地点
mark	varchar(200)	否	备注

☑ tb_inaccount（收入信息表）

tb_inaccount 表用于保存用户的收入信息，该表的结构如表 21.8 所示。

表 21.8 收入信息表

字 段 名	数 据 类 型	主 键 否	描 述
_id	integer	是	编号
money	decimal	否	收入金额
time	varchar(10)	否	收入时间
type	varchar(10)	否	收入类别





续表

字　段　名	数 据 类 型	主　键　否	描　　述
handler	varchar(100)	否	付款方
mark	varchar(200)	否	备注

☑　tb_flag（便签信息表）

tb_flag 表用于保存家庭理财通的便签信息，该表的结构如表 21.9 所示。

表 21.9　便签信息表

字　段　名	数 据 类 型	主　键　否	描　　述
_id	integer	是	编号
flag	varchar(200)	否	便签内容

21.5　系统文件夹组织结构

在编写项目代码之前，需要制定好项目的系统文件夹组织结构，如不同的 Java 包存放不同的窗体、公共类、数据模型、工具类或者图片资源等，这样不但可以保证团队开发的一致性，也可以规范系统的整体架构。创建完系统中可能用到的文件夹或者 Java 包之后，在开发时，只需将创建的类文件或者资源文件保存到相应的文件夹中即可。家庭理财通系统的文件夹组织结构如图 21.4 所示。

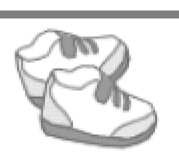


图 21.4　文件夹组织结构



说明：

从图 21.4 中可以看到，res 和 assets 文件夹都用来存放资源文件，但在实际开发时，Android 不为 assets 文件夹下的资源文件生成 ID，用户需要通过 AssetManager 类以文件路径和文件名的方式来访问 assets 文件夹中的文件。





Note

21.6　公共类设计

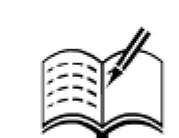
公共类是代码重用的一种形式，它将各个功能模块经常调用的方法提取到公用的 Java 类中，例如，访问数据库的 Dao 类容纳了所有访问数据库的方法，并同时管理着数据库的连接、关闭等内容。使用公共类，不但实现了项目代码的重用，还提供了程序的性能和代码的可读性。本节将介绍家庭理财通中的公共类设计。

21.6.1　数据模型公共类

在 com.xiaoke.accountsoft.model 包中存放的是数据模型公共类，它们对应着数据库中不同的数据表，这些模型将被访问数据库的 Dao 类和程序中各个模块甚至各个组件所使用。数据模型是对数据表中所有字段的封装，它主要用于存储数据，并通过相应的 getXXX()方法和 setXXX()方法实现不同属性的访问原则。现在以收入信息表为例，介绍它所对应的数据模型类的实现代码，其主要代码如下：

```
package com.xiaoke.accountsoft.model;
public class Tb_inaccount                                   //收入信息实体类
{
	private int _id;                                        //存储收入编号
	private double money;                                   //存储收入金额
	private String time;                                    //存储收入时间
	private String type;                                    //存储收入类别
	private String handler;                                 //存储收入付款方
	private String mark;                                    //存储收入备注
	public Tb_inaccount()                                   //默认构造函数
	{
		super();
	}
	//定义有参构造函数，用来初始化收入信息实体类中的各个字段
	public Tb_inaccount(int id, double money, String time,String type,String handler,String mark)
	{
		super();
		this._id = id;                                      //为收入编号赋值
		this.money = money;                                 //为收入金额赋值
		this.time = time;                                   //为收入时间赋值
		this.type = type;                                   //为收入类别赋值
		this.handler = handler;                             //为收入付款方赋值
		this.mark = mark;                                   //为收入备注赋值
	}
	public int getid()                                      //设置收入编号的可读属性
	{
		return _id;
```

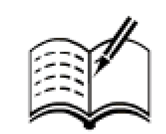





```
        }
        public void setid(int id)                    //设置收入编号的可写属性
        {
            this._id = id;
        }
        public double getMoney()                     //设置收入金额的可读属性
        {
            return money;
        }
        public void setMoney(double money)           //设置收入金额的可写属性
        {
            this.money = money;
        }
        public String getTime()                      //设置收入时间的可读属性
        {
            return time;
        }
        public void setTime(String time)             //设置收入时间的可写属性
        {
            this.time = time;
        }
        public String getType()                      //设置收入类别的可读属性
        {
            return type;
        }
        public void setType(String type)             //设置收入类别的可写属性
        {
            this.type = type;
        }
        public String getHandler()                   //设置收入付款方的可读属性
        {
            return handler;
        }
        public void setHandler(String handler)       //设置收入付款方的可写属性
        {
            this.handler = handler;
        }
        public String getMark()                      //设置收入备注的可读属性
        {
            return mark;
        }
        public void setMark(String mark)             //设置收入备注的可写属性
        {
            this.mark = mark;
        }
}
```

其他数据模型类的定义与收入数据模型类的定义方法类似，其属性内容就是数据表中相应的字段。




Note

com.xiaoke.accountsoft.model 包中包含的数据模型类如表 21.10 所示。

表 21.10　com.xiaoke.accountsoft.model 包中的数据模型类

类　　名	说　　明
Tb_flag	便签信息数据表模型类
Tb_inaccount	收入信息数据表模型类
Tb_outaccount	支出信息数据表模型类
Tb_pwd	密码信息数据表模型类

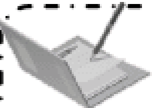

说明：

表 21.10 中的所有模型类都定义了对应数据表字段的属性，并提供了访问相应属性的 getXXX() 和 setXXX()方法。

21.6.2　Dao 公共类

Dao 的全称是 Data Access Object，即数据访问对象，本系统中创建了 com.xiaoke.accountsoft.dao 包，该包中包含了 DBOpenHelper、FlagDAO、InaccountDAO、OutaccountDAO 和 PwdDAO 5 个数据访问类，其中，DBOpenHelper 类用来实现创建数据库、数据表等功能；FlagDAO 类用来对便签信息进行管理；InaccountDAO 类用来对收入信息进行管理；OutaccountDAO 类用来对支出信息进行管理；PwdDAO 类用来对密码信息进行管理。下面主要对 DBOpenHelper 类和 InaccountDAO 类进行详细讲解。

说明：

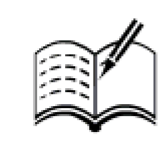
FlagDAO、OutaccountDAO 和 PwdDAO 类的实现过程，与 InaccountDAO 类类似，这里不再进行详细介绍，详请参见本书附带光盘中的源代码。

1．DBOpenHelper.java 类

DBOpenHelper 类主要用来实现创建数据库和数据表的功能，该类继承自 SQLiteOpenHelper 类，在该类中，首先需要在构造函数中创建数据库，然后在覆写的 onCreate()方法中使用 SQLiteDatabase 对象的 execSQL()方法分别创建 tb_outaccount、tb_inaccount、tb_pwd 和 tb_flag 4 个数据表。DBOpenHelper 类的实现代码如下：

```
package com.xiaoke.accountsoft.dao;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
public class DBOpenHelper extends SQLiteOpenHelper
{
    private static final int VERSION = 1;                          //定义数据库版本号
    private static final String DBNAME = "account.db";             //定义数据库名
```






```
    public DBOpenHelper(Context context)                          //定义构造函数
    {
        super(context, DBNAME, null, VERSION);                    //重写基类的构造函数
    }
    @Override
    public void onCreate(SQLiteDatabase db)                       //创建数据库
    {
        //创建支出信息表
        db.execSQL("create table tb_outaccount (_id integer primary key,money decimal,time
varchar(10)," +"type varchar(10),address varchar(100),mark varchar(200))");
        //创建收入信息表
        db.execSQL("create table tb_inaccount (_id integer primary key,money decimal,time varchar(10)," +
                "type varchar(10),handler varchar(100),mark varchar(200))");
        db.execSQL("create table tb_pwd (password varchar(20))");    //创建密码表
        //创建便签信息表
        db.execSQL("create table tb_flag (_id integer primary key,flag varchar(200))");
    }
    //覆写基类的 onUpgrade()方法，以便数据库版本更新
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    {
    }
}
```

2. InaccountDAO.java 类

InaccountDAO 类主要用来对收入信息进行管理，包括收入信息的添加、修改、删除、查询及获取最大编号、总记录数等功能，下面对该类中的方法进行详细讲解。

☑ InaccountDAO 类的构造函数

在 InaccountDAO 类中定义两个对象，分别是 DBOpenHelper 和 SQLiteDatabase 对象，然后创建该类的构造函数，在构造函数中初始化 DBOpenHelper 对象。其主要代码如下：

```
private DBOpenHelper helper;                              //创建 DBOpenHelper 对象
private SQLiteDatabase db;                                //创建 SQLiteDatabase 对象
public InaccountDAO(Context context)                      //定义构造函数
{
    helper = new DBOpenHelper(context);                   //初始化 DBOpenHelper 对象
}
```

☑ add(Tb_inaccount tb_inaccount)方法

add(Tb_inaccount tb_inaccount)方法的主要功能是添加收入信息，其中，参数 tb_inaccount 表示收入数据表对象。其主要代码如下：

```
/**
 * 添加收入信息
 *
 * @param tb_inaccount
 */
```

```
public void add(Tb_inaccount tb_inaccount)
{
    db = helper.getWritableDatabase();                                    //初始化 SQLiteDatabase 对象
    //执行添加收入信息操作
    db.execSQL("insert into tb_inaccount (_id,money,time,type,handler,mark) values (?,?,?,?,?,?)", new
Object[]
    {tb_inaccount.getid(),tb_inaccount.getMoney(), tb_inaccount.getTime(),tb_inaccount.getType(), tb_inaccount.
getHandler(),tb_inaccount.getMark() });
}
```

☑ update(Tb_inaccount tb_inaccount)方法

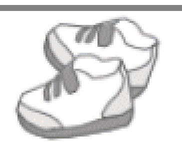
update(Tb_inaccount tb_inaccount)方法的主要功能是根据指定的编号修改收入信息，其中，参数 tb_inaccount 表示收入数据表对象。其主要代码如下：

```
/**
 * 更新收入信息
 *
 * @param tb_inaccount
 */
public void update(Tb_inaccount tb_inaccount)
{
    db = helper.getWritableDatabase();                                    //初始化 SQLiteDatabase 对象
    //执行修改收入信息操作
    db.execSQL("update tb_inaccount set money = ?,time = ?,type = ?,handler = ?,mark = ? where _id
= ?", new Object[]
    {tb_inaccount.getMoney(), tb_inaccount.getTime(),tb_inaccount.getType(),tb_inaccount.getHandler(),
tb_inaccount.getMark(),tb_inaccount.getid() });
}
```

☑ find(int id)方法

find(int id)方法的主要功能是根据指定的编号查找收入信息，其中，参数 id 表示要查找的收入编号，返回值为 Tb_inaccount 对象。其主要代码如下：

```
/**
 * 查找收入信息
 *
 * @param id
 * @return
 */
public Tb_inaccount find(int id)
{
    db = helper.getWritableDatabase();                  //初始化 SQLiteDatabase 对象
    Cursor cursor = db.rawQuery("select _id,money,time,type,handler,mark from tb_inaccount where
_id = ?", new String[]
    { String.valueOf(id) });                            //根据编号查找收入信息，并存储到 Cursor 类中
    if (cursor.moveToNext())                            //遍历查找到的收入信息
    {
        //将遍历到的收入信息存储到 Tb_inaccount 类中
```



```
            return new Tb_inaccount(cursor.getInt(cursor.getColumnIndex("_id")), cursor.getDouble(cursor.
getColumnIndex ("money")), cursor.getString(cursor.getColumnIndex("time")), cursor.getString(cursor. getColumnIndex
("type")), cursor.getString(cursor.getColumnIndex("handler")), cursor.getString(cursor. getColumnIndex
("mark")));
        }
        return null;                                          //如果没有信息，则返回 null
}
```

☑　detele(Integer... ids)方法

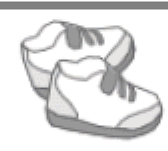
detele(Integer... ids)方法的主要功能是根据指定的一系列编号删除收入信息，其中，参数 ids 表示要删除的收入编号的集合。其主要代码如下：

```
/**
 * 删除收入信息
 *
 * @param ids
 */
public void detele(Integer... ids)
{
    if (ids.length > 0)                                     //判断是否存在要删除的 id
    {
        StringBuffer sb = new StringBuffer();               //创建 StringBuffer 对象
        for (int i = 0; i < ids.length; i++)                //遍历要删除的 id 集合
        {
            sb.append('?').append(',');                     //将删除条件添加到 StringBuffer 对象中
        }
        sb.deleteCharAt(sb.length() - 1);                   //去掉最后一个“,”字符
        db= helper.getWritableDatabase();                   //初始化 SQLiteDatabase 对象
        //执行删除收入信息操作
        db.execSQL("delete from tb_inaccount where _id in (" + sb + ")", (Object[]) ids);
    }
}
```

☑　getScrollData(int start, int count)方法

getScrollData(int start, int count)方法的主要功能是从收入数据表的指定索引处获取指定数量的收入数据，其中，参数 start 表示要从此处开始获取数据的索引；参数 count 表示要获取的数量，返回值为 List<Tb_inaccount>对象。其主要代码如下：

```
/**
 * 获取收入信息
 * @param start 起始位置
 * @param count 每页显示数量
 * @return
 */
public List<Tb_inaccount> getScrollData(int start, int count)
{
    List<Tb_inaccount> tb_inaccount = new ArrayList<Tb_inaccount>(); //创建集合对象
    db = helper.getWritableDatabase();                               //初始化 SQLiteDatabase 对象
    Cursor cursor = db.rawQuery("select * from tb_inaccount limit ?,?", new String[]{ String.valueOf
(start), String.valueOf(count) });                                   //获取所有收入信息
```

Note

```
        while (cursor.moveToNext())                                   //遍历所有的收入信息
        {
            tb_inaccount.add(new Tb_inaccount(cursor.getInt(cursor.getColumnIndex("_id")), cursor.getDouble
(cursor.getColumnIndex("money")), cursor.getString(cursor.getColumnIndex("time")), cursor.getString(cursor.
getColumnIndex("type")), cursor.getString(cursor.getColumnIndex("handler")), cursor.getString(cursor.
getColumnIndex("mark"))));                                  //将遍历到的收入信息添加到集合中
        }
        return tb_inaccount;                                          //返回集合
}
```

☑ getCount()方法

getCount()方法的主要功能是获取收入数据表中的总记录数，返回值为获取到的总记录数。其主要代码如下：

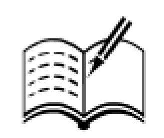
```
/**
 * 获取总记录数
 * @return
 */
public long getCount()
{
        db = helper.getWritableDatabase();                            //初始化 SQLiteDatabase 对象
        //获取收入信息的记录数
        Cursor cursor = db.rawQuery("select count(_id) from tb_inaccount", null);
        if (cursor.moveToNext())                                      //判断 Cursor 中是否有数据
        {
            return cursor.getLong(0);                                 //返回总记录数
        }
        return 0;                                                     //如果没有数据，则返回 0
}
```

☑ getMaxId()方法

getMaxId()方法的主要功能是获取收入数据表中的最大编号，返回值为获取到的最大编号。其主要代码如下：

```
/**
 * 获取收入最大编号
 * @return
 */
public int getMaxId()
{
        db = helper.getWritableDatabase();                            //初始化 SQLiteDatabase 对象
        //获取收入信息表中的最大编号
        Cursor cursor = db.rawQuery("select max(_id) from tb_inaccount", null);
        while (cursor.moveToLast()) {                                 //访问 Cursor 中的最后一条数据
            return cursor.getInt(0);                                  //获取访问到的数据，即最大编号
        }
        return 0;                                                     //如果没有数据，则返回 0
}
```





21.7　登录模块设计

登录模块主要是通过输入正确的密码进入家庭理财通的主窗体，它可以提高程序的安全性，保护数据资料不外泄。登录模块的运行效果如图 21.5 所示。

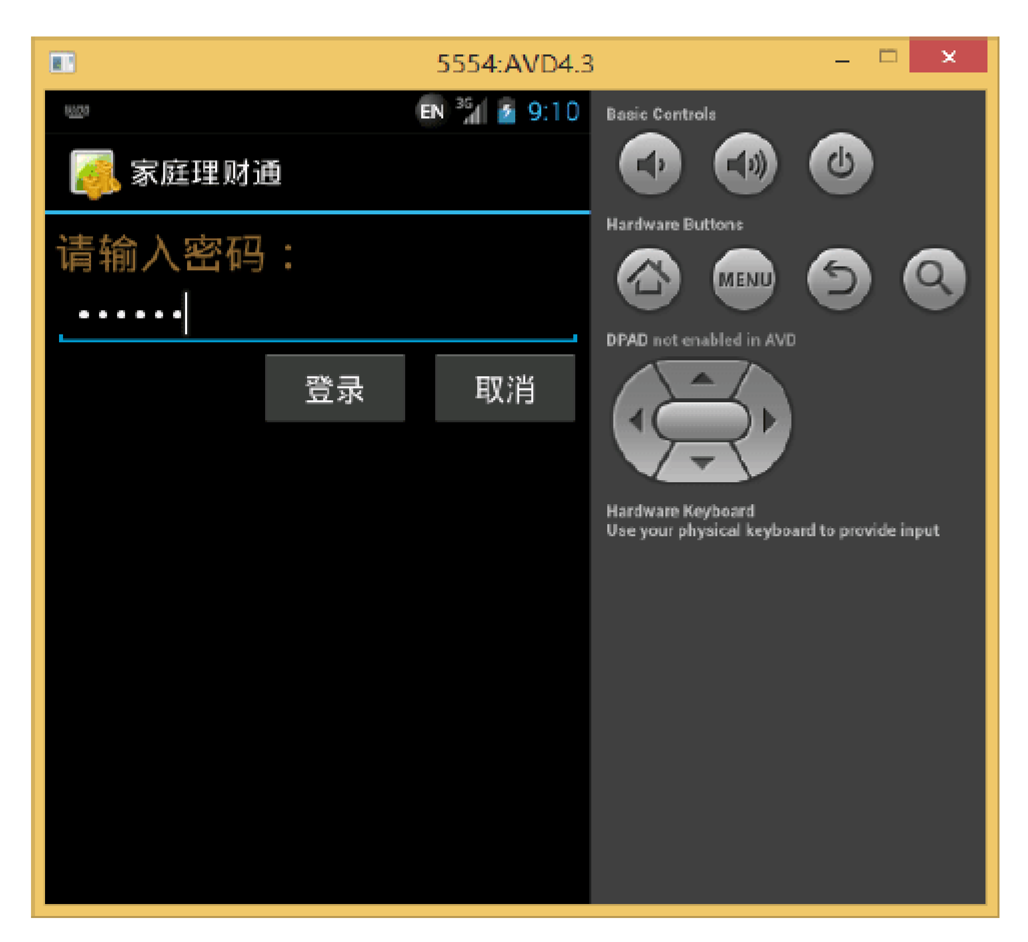
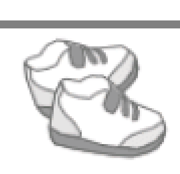


图 21.5　系统登录

21.7.1　设计登录布局文件

在 res\layout 目录下新建一个 login.xml 文件，用来作为登录窗体的布局文件，在该布局文件中，将布局方式修改为 RelativeLayout，然后添加一个 TextView 组件、一个 EditText 组件和两个 Button 组件。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dp"
    >
    <TextView android:id="@+id/tvLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:text="请输入密码："
        android:textSize="25dp"
        android:textColor="#8C6931"
    />
```



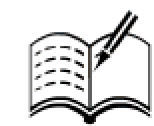

```
    <EditText android:id="@+id/txtLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/tvLogin"
        android:inputType="textPassword"
        android:hint="请输入密码"
    />
    <Button android:id="@+id/btnClose"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtLogin"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dp"
        android:text="取消"
    />
    <Button android:id="@+id/btnLogin"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtLogin"
        android:layout_toLeftOf="@id/btnClose"
        android:text="登录"
    />
</RelativeLayout>
```

21.7.2 登录功能的实现

在 com.xiaoke.accountsoft.activity 包中创建一个 Login.java 文件，该文件的布局文件设置为 login.xml。当用户在“请输入密码”文本框中输入密码时，单击“登录”按钮，为“登录”按钮设置监听事件，在监听事件中，判断数据库中是否设置了密码、输入的密码为空，或者输入的密码是否与数据库中的密码一致，如果条件满足，则登录主 Activity；否则，弹出信息提示框。其代码如下：

```
txtlogin=(EditText) findViewById(R.id.txtLogin);                //获取“密码”文本框
btnlogin=(Button) findViewById(R.id.btnLogin);                  //获取“登录”按钮
btnlogin.setOnClickListener(new OnClickListener() {             //为“登录”按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        Intent intent=new Intent(Login.this, MainActivity.class);     //创建 Intent 对象
        PwdDAO pwdDAO=new PwdDAO(Login.this);                         //创建 PwdDAO 对象
        if((pwdDAO.getCount()==0|  pwdDAO.find().getPassword().isEmpty())  &&  txtlogin.getText().
toString(). isEmpty()){                                               //判断是否有密码及是否输入了密码
            startActivity(intent);                                    //启动主 Activity
        }
```





Note

```
            else {
                //判断输入的密码是否与数据库中的密码一致
                if (pwdDAO.find().getPassword().equals(txtlogin.getText().toString())) {
                    startActivity(intent);                              //启动主 Activity
                }
                else {
                    //弹出信息提示
                    Toast.makeText(Login.this, "请输入正确的密码！", Toast.LENGTH_SHORT).show();
                }
            }
            txtlogin.setText("");                                       //清空“密码”文本框
        }
});
```



说明：

本系统在 com.xiaoke.accountsoft.activity 包中创建的.java 类文件，都是基于 Activity 类的，下面遇到时，将不再说明。

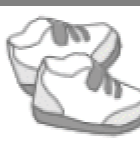
21.7.3 退出登录窗口

单击“取消”按钮，为“取消”按钮设置监听事件，在监听事件中调用 finish()方法实现退出当前程序的功能。其代码如下：

```
btnclose=(Button) findViewById(R.id.btnClose);                  //获取“取消”按钮
btnclose.setOnClickListener(new OnClickListener() {             //为“取消”按钮设置监听事件
        @Override
        public void onClick(View arg0) {
            //TODO Auto-generated method stub
            finish();                                           //退出当前程序
        }
});
```

21.8 系统主窗体设计

主窗体是程序操作过程中必不可少的，它是与用户交互的重要环节。通过主窗体，用户可以调用系统相关的各子模块，快速掌握本系统中所实现的各个功能。在家庭理财通系统中，当登录窗体验证成功后，用户将进入主窗体，主窗体中以图标和文本相结合的方式显示各功能按钮，单击这些功能按钮时，打开相应功能的 Activity。主窗体的运行效果如图 21.6 所示。

Note



图 21.6　家庭理财通主窗体

21.8.1　设计系统主窗体布局文件

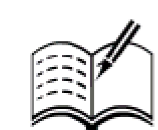
在 res\layout 目录下新建一个 main.xml 文件，用来作为主窗体的布局文件，在该布局文件中，添加一个 GridView 组件，用来显示功能图标及文本。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/gvInfo"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:columnWidth="90dp"
        android:numColumns="auto_fit"
        verticalSpacing="10dp"
        android:horizontalSpacing="10dp"
        android:stretchMode="spacingWidthUniform"
        android:gravity="center"
/>
```

在 res\layout 目录下再新建一个 gvitem.xml 文件，用来为 main.xml 布局文件中的 GridView 组件提供资源，在该文件中添加一个 ImageView 组件和一个 TextView 组件。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/item"
        android:orientation="vertical"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        >
```






```
    <ImageView android:id="@+id/ItemImage"
        android:layout_width="75dp"
        android:layout_height="75dp"
        android:layout_gravity="center"
        android:scaleType="fitXY"
        android:padding="4dp"
    />
    <TextView android:id="@+id/ItemTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
    />
</LinearLayout>
```

21.8.2 显示各功能窗口

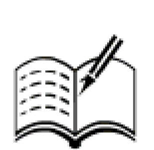
在 com.xiaoke.accountsoft.activity 包中创建一个 MainActivity.java 文件，该文件的布局文件设置为 main.xml。在 MainActivity.java 文件中，首先创建一个 GridView 组件对象，然后分别定义一个 String 类型的数组和一个 int 类型的数组，它们分别用来存储系统功能的文本及对应的图标。其代码如下：

```
GridView gvInfo;                                    //创建 GridView 对象
String[] titles=new String[]{"新增支出","新增收入","我的支出","我的收入","数据管理","系统设置","收支便签
","退出"};                                          //定义字符串数组，存储系统功能
int[] images=new int[]{R.drawable.addoutaccount,R.drawable.addinaccount,
    R.drawable.outaccountinfo,R.drawable.inaccountinfo,R.drawable.showinfo,R.drawable.sysset,
R.drawable.accountflag,R.drawable.exit};            //定义 int 数组，存储功能对应的图标
```

当用户在主窗体中单击各功能按钮时，使用相应功能所对应的 Activity 初始化 Intent 对象，然后使用 startActivity()方法启动相应的 Activity，而如果用户单击的是“退出”功能按钮，则调用 finish()方法关闭当前 Activity。其代码如下：

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    gvInfo=(GridView) findViewById(R.id.gvInfo);        //获取布局文件中的 gvInfo 组件
    //创建 pictureAdapter 对象
    pictureAdapter adapter=new pictureAdapter(titles,images,this);
    gvInfo.setAdapter(adapter);                         //为 GridView 设置数据源
    gvInfo.setOnItemClickListener(new OnItemClickListener() {      //为 GridView 设置项单击事件
        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3) {
            Intent intent = null;                       //创建 Intent 对象
            switch (arg2) {
```



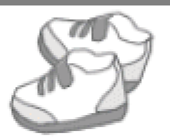



```
                case 0:
                //使用 AddOutaccount 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, AddOutaccount.class);
                    startActivity(intent);                    //打开 AddOutaccount
                    break;
                case 1:
                //使用 AddInaccount 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, AddInaccount.class);
                    startActivity(intent);                    //打开 AddInaccount
                    break;
                case 2:
                //使用 Outaccountinfo 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, Outaccountinfo.class);
                    startActivity(intent);                    //打开 Outaccountinfo
                    break;
                case 3:
                //使用 Inaccountinfo 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, Inaccountinfo.class);
                    startActivity(intent);                    //打开 Inaccountinfo
                    break;
                case 4:
                //使用 Showinfo 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, Showinfo.class);
                    startActivity(intent);                    //打开 Showinfo
                    break;
                case 5:
                //使用 Sysset 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, Sysset.class);
                    startActivity(intent);                    //打开 Sysset
                    break;
                case 6:
                //使用 Accountflag 窗口初始化 Intent
                    intent=new Intent(MainActivity.this, Accountflag.class);
                    startActivity(intent);                    //打开 Accountflag
                    break;
                case 7:
                    finish();                                 //关闭当前 Activity
                }
            }
        });
}
```

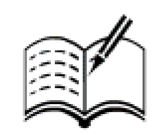
21.8.3 定义文本及图片组件

定义一个 ViewHolder 类，用来定义文本组件及图片组件对象。其代码如下：

```
class ViewHolder                                    //创建 ViewHolder 类
{
```



```
    public TextView title;                              //创建 TextView 对象
    public ImageView image;                             //创建 ImageView 对象
}
```



21.8.4 定义功能图标及说明文字

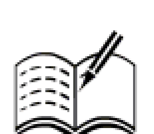
定义一个 Picture 类，用来定义功能图标及说明文字的实体。其代码如下：

```
class Picture                                           //创建 Picture 类
{
    private String title;                               //定义字符串，表示图像标题
    private int imageId;                                //定义 int 变量，表示图像的二进制值
    public Picture()                                    //默认构造函数
    {
        super();
    }
    public Picture(String title,int imageId)            //定义有参构造函数
    {
        super();
        this.title=title;                               //为图像标题赋值
        this.imageId=imageId;                           //为图像的二进制值赋值
    }
    public String getTitle() {                          //定义图像标题的可读属性
        return title;
    }
    public void setTitle(String title) {                //定义图像标题的可写属性
        this.title=title;
    }
    public int getImageId() {                           //定义图像二进制值的可读属性
        return imageId;
    }
    public void setimageId(int imageId) {               //定义图像二进制值的可写属性
        this.imageId=imageId;
    }
}
```

21.8.5 设置功能图标及说明文字

定义一个 pictureAdapter 类，该类继承自 BaseAdapter 类，该类用来分别为 ViewHolder 类中的 TextView 和 ImageView 组件设置功能的说明性文字及图标。其代码如下：

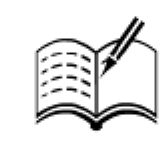
```
class pictureAdapter extends BaseAdapter               //创建基于 BaseAdapter 的子类
{
    private LayoutInflater inflater;                    //创建 LayoutInflater 对象
    private List<Picture> pictures;                     //创建 List 泛型集合
```


Note

```
//为类创建构造函数
public pictureAdapter(String[] titles,int[] images,Context context) {
    super();
    pictures=new ArrayList<Picture>();                //初始化泛型集合对象
    inflater=LayoutInflater.from(context);            //初始化 LayoutInflater 对象
    for(int i=0;i<images.length;i++)                  //遍历图像数组
    {
        Picture picture=new Picture(titles[i], images[i]); //使用标题和图像生成 Picture
        pictures.add(picture);                        //将 Picture 对象添加到泛型集合中
    }
}
@Override
public int getCount() {                               //获取泛型集合的长度
    //TODO Auto-generated method stub
    if (null != pictures) {                           //如果泛型集合不为空
        return pictures.size();                       //返回泛型长度
    }
    else {
        return 0;//返回 0
    }
}
@Override
public Object getItem(int arg0) {
    //TODO Auto-generated method stub
    return pictures.get(arg0);                        //获取泛型集合指定索引处的项
}
@Override
public long getItemId(int arg0) {
    //TODO Auto-generated method stub
    return arg0;                                      //返回泛型集合的索引
}
@Override
public View getView(int arg0, View arg1, ViewGroup arg2) {
    //TODO Auto-generated method stub
    ViewHolder viewHolder;                            //创建 ViewHolder 对象
    if(arg1==null)                                    //判断图像标识是否为空
    {
        arg1=inflater.inflate(R.layout.gvitem, null); //设置图像标识
        viewHolder=new ViewHolder();//初始化 ViewHolder 对象
        viewHolder.title=(TextView) arg1.findViewById(R.id.ItemTitle);//设置图像标题
        //设置图像的二进制值
        viewHolder.image=(ImageView) arg1.findViewById(R.id.ItemImage);
        arg1.setTag(viewHolder);                      //设置提示
    }
    else {
        viewHolder=(ViewHolder) arg1.getTag();        //设置提示
    }
    viewHolder.title.setText(pictures.get(arg0).getTitle()); //设置图像标题
    //设置图像的二进制值
    viewHolder.image.setImageResource(pictures.get(arg0).getImageId());
```



```
            return   arg1;                                  //返回图像标识
        }
}
```



Note

21.9　收入管理模块设计

收入管理模块主要包括 3 部分，分别是“新增收入”、“收入信息浏览”和“修改/删除收入信息”，其中，“新增收入”用来添加收入信息；“收入信息浏览”用来显示所有的收入信息；“修改/删除收入信息”用来根据编号修改或者删除收入信息。本节将从这 3 个方面对收入管理模块进行详细介绍。

首先来看新增收入模块，新增收入窗体运行效果如图 21.7 所示。

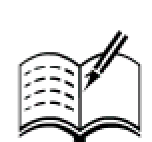


图 21.7　新增收入

21.9.1　设计新增收入布局文件

在 res\layout 目录下新建一个 addinaccount.xml 文件，用来作为新增收入窗体的布局文件，该布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加 5 个 TextView 组件、4 个 EditText 组件、1 个 Spinner 组件和 2 个 Button 组件。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/initem"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```

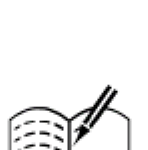

Note

```
<LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="3"
      >
      <TextView
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:gravity="center_horizontal"
            android:text="新增收入"
            android:textSize="40sp"
            android:textColor="#ffffff"
            android:textStyle="bold"
            android:layout_height="wrap_content"/>
</LinearLayout>
<LinearLayout
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      >
      <RelativeLayout android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:padding="10dp"
           >
           <TextView android:layout_width="90dp"
           android:id="@+id/tvInMoney"
           android:textSize="20sp"
           android:text="金  额："
           android:layout_height="wrap_content"
           android:layout_alignBaseline="@+id/txtInMoney"
           android:layout_alignBottom="@+id/txtInMoney"
           android:layout_alignParentLeft="true"
           android:layout_marginLeft="16dp">
           </TextView>
           <EditText
           android:id="@+id/txtInMoney"
           android:layout_width="210dp"
           android:layout_height="wrap_content"
           android:layout_toRightOf="@id/tvInMoney"
           android:inputType="number"
           android:numeric="integer"
           android:maxLength="9"
           android:hint="0.00"
           />
           <TextView android:layout_width="90dp"
           android:id="@+id/tvInTime"
           android:textSize="20sp"
```






```
android:text="时　间："
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/txtInTime"
android:layout_alignBottom="@+id/txtInTime"
android:layout_toLeftOf="@+id/txtInMoney">
</TextView>
<EditText
android:id="@+id/txtInTime"
android:layout_width="210dp"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/tvInTime"
android:layout_below="@id/txtInMoney"
android:inputType="datetime"
android:hint="2011-01-01"
/>
<TextView android:layout_width="90dp"
android:id="@+id/tvInType"
android:textSize="20sp"
android:text="类　别："
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/spInType"
android:layout_alignBottom="@+id/spInType"
android:layout_alignLeft="@+id/tvInTime">
</TextView>
<Spinner android:id="@+id/spInType"
android:layout_width="210dp"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/tvInType"
android:layout_below="@id/txtInTime"
android:entries="@array/intype"
/>
<TextView android:layout_width="90dp"
android:id="@+id/tvInHandler"
android:textSize="20sp"
android:text="付款方："
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/txtInHandler"
android:layout_alignBottom="@+id/txtInHandler"
android:layout_toLeftOf="@+id/spInType">
</TextView>
<EditText
android:id="@+id/txtInHandler"
android:layout_width="210dp"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/tvInHandler"
android:layout_below="@id/spInType"
android:singleLine="false"
/>
```


Note

```
                <TextView android:layout_width="90dp"
                android:id="@+id/tvInMark"
                android:textSize="20sp"
                android:text="备　注："
                android:layout_height="wrap_content"
                android:layout_alignTop="@+id/txtInMark"
                android:layout_toLeftOf="@+id/txtInHandler">
                </TextView>
                <EditText
                android:id="@+id/txtInMark"
                android:layout_width="210dp"
                android:layout_height="150dp"
                android:layout_toRightOf="@id/tvInMark"
                android:layout_below="@id/txtInHandler"
                android:gravity="top"
                android:singleLine="false"
                />
                </RelativeLayout>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="3"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="10dp"
            >
        <Button
            android:id="@+id/btnInCancel"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dp"
            android:text="取消"
            />
            <Button
             android:id="@+id/btnInSave"
             android:layout_width="80dp"
             android:layout_height="wrap_content"
             android:layout_toLeftOf="@id/btnInCancel"
             android:text="保存"
            />
        </RelativeLayout>
    </LinearLayout>
</LinearLayout>
```



21.9.2 设置收入时间

在 com.xiaoke.accountsoft.activity 包中创建一个 AddInaccount.java 文件，该文件的布局文件设置为 addinaccount.xml。在 AddInaccount.java 文件中，首先创建类中需要用到的全局对象及变量。其代码如下：

```
protected static final int DATE_DIALOG_ID = 0;            //创建日期对话框常量
EditText txtInMoney,txtInTime,txtInHandler,txtInMark;     //创建 4 个 EditText 对象
Spinner spInType;                                         //创建 Spinner 对象
Button btnInSaveButton;                                   //创建 Button 对象“保存”
Button btnInCancelButton;                                 //创建 Button 对象“取消”
private int mYear;                                        //年
private int mMonth;                                       //月
private int mDay;                                         //日
```

在 onCreate()覆写方法中，初始化创建的 EidtText、Spinner 和 Button。其代码如下：

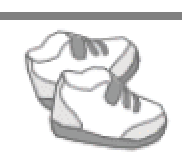
```
txtInMoney=(EditText) findViewById(R.id.txtInMoney);           //获取“金额”文本框
txtInTime=(EditText) findViewById(R.id.txtInTime);             //获取“时间”文本框
txtInHandler=(EditText) findViewById(R.id.txtInHandler);       //获取“付款方”文本框
txtInMark=(EditText) findViewById(R.id.txtInMark);             //获取“备注”文本框
spInType=(Spinner) findViewById(R.id.spInType);                //获取“类别”下拉列表
btnInSaveButton=(Button) findViewById(R.id.btnInSave);         //获取“保存”按钮
btnInCancelButton=(Button) findViewById(R.id.btnInCancel);     //获取“取消”按钮
```

单击“时间”文本框，为该文本框设置监听事件，在监听事件中使用 showDialog()方法弹出时间选择对话框；并且在 Activity 创建时，默认显示当前的系统时间。其代码如下：

```
txtInTime.setOnClickListener(new OnClickListener() {      //为“时间”文本框设置单击监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        showDialog(DATE_DIALOG_ID);                       //显示日期选择对话框
    }
});
final Calendar c = Calendar.getInstance();                //获取当前系统日期
mYear = c.get(Calendar.YEAR);                             //获取年份
mMonth = c.get(Calendar.MONTH);                           //获取月份
mDay = c.get(Calendar.DAY_OF_MONTH);                      //获取天数
updateDisplay();                                          //显示当前系统时间
```

上面的代码中用到了 updateDisplay()方法，该方法用来显示设置的时间。其其代码如下：

```
private void updateDisplay()
{
    txtInTime.setText(new  StringBuilder().append(mYear).append("-").append(mMonth  +  1).append  ("-").
append (mDay));                                           //显示设置的时间
}
```



Note

在为“时间”文本框设置监听事件时，弹出了时间选择对话框，该对话框的弹出需要覆写onCreateDialog()方法，该方法用来根据指定的标识弹出时间选择对话框。其代码如下：

```
@Override
protected Dialog onCreateDialog(int id)                          //重写 onCreateDialog()方法
{
    switch (id)
    {
    case DATE_DIALOG_ID:                                         //弹出时间选择对话框
        return new DatePickerDialog(this, mDateSetListener, mYear, mMonth, mDay);
    }
    return null;
}
```

上面的代码中用到了 mDateSetListener 对象，该对象是 OnDateSetListener 类的一个对象，用来显示用户设置的时间。其代码如下：

```
private DatePickerDialog.OnDateSetListener mDateSetListener = new DatePickerDialog.OnDateSetListener()
{
    public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth)
    {
        mYear = year;                                            //为年份赋值
        mMonth = monthOfYear;                                    //为月份赋值
        mDay = dayOfMonth;                                       //为天赋值
        updateDisplay();                                         //显示设置的日期
    }
};
```

21.9.3 添加收入信息

填写完信息后，单击“保存”按钮，为该按钮设置监听事件，在监听事件中，使用 InaccountDAO 对象的 add()方法将用户的输入保存到收入信息表中。其代码如下：

```
btnInSaveButton.setOnClickListener(new OnClickListener() {          //为“保存”按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        String strInMoney= txtInMoney.getText().toString();         //获取“金额”文本框的值
        if(!strInMoney.isEmpty()){                                   //判断金额不为空
            //创建 InaccountDAO 对象
            InaccountDAO inaccountDAO=new InaccountDAO(AddInaccount.this);
            Tb_inaccount tb_inaccount=new Tb_inaccount(inaccountDAO.getMaxId()+1, Double.parseDouble
(strInMoney), txtInTime.getText().toString(), spInType.getSelectedItem().toString(), txtInHandler.getText().
toString(), txtInMark.getText().toString());                        //创建 Tb_inaccount 对象
            inaccountDAO.add(tb_inaccount);                          //添加收入信息
            //弹出信息提示
            Toast.makeText(AddInaccount.this, "〖新增收入〗数据添加成功！",Toast.LENGTH_SHORT). show();
```



```
            }
            else {
                Toast.makeText(AddInaccount.this, "请输入收入金额！",Toast.LENGTH_SHORT).show();
            }
        }
});
```



Note

21.9.4 重置新增收入窗体中的各个控件

单击“取消”按钮，重置新增收入窗体中的各个控件。其代码如下：

```
btnInCancelButton.setOnClickListener(new OnClickListener() {//为“取消”按钮设置监听事件
        @Override
        public void onClick(View arg0) {
            //TODO Auto-generated method stub
            txtInMoney.setText("");                    //设置“金额”文本框为空
            txtInMoney.setHint("0.00");                //为“金额”文本框设置提示
            txtInTime.setText("");                     //设置“时间”文本框为空
            txtInTime.setHint("2011-01-01");           //为“时间”文本框设置提示
            txtInHandler.setText("");                  //设置“付款方”文本框为空
            txtInMark.setText("");                     //设置“备注”文本框为空
            spInType.setSelection(0);                  //设置“类别”下拉列表默认选择第一项
        }
});
```

21.9.5 设计收入信息浏览布局文件

收入信息浏览窗体的运行效果如图 21.8 所示。

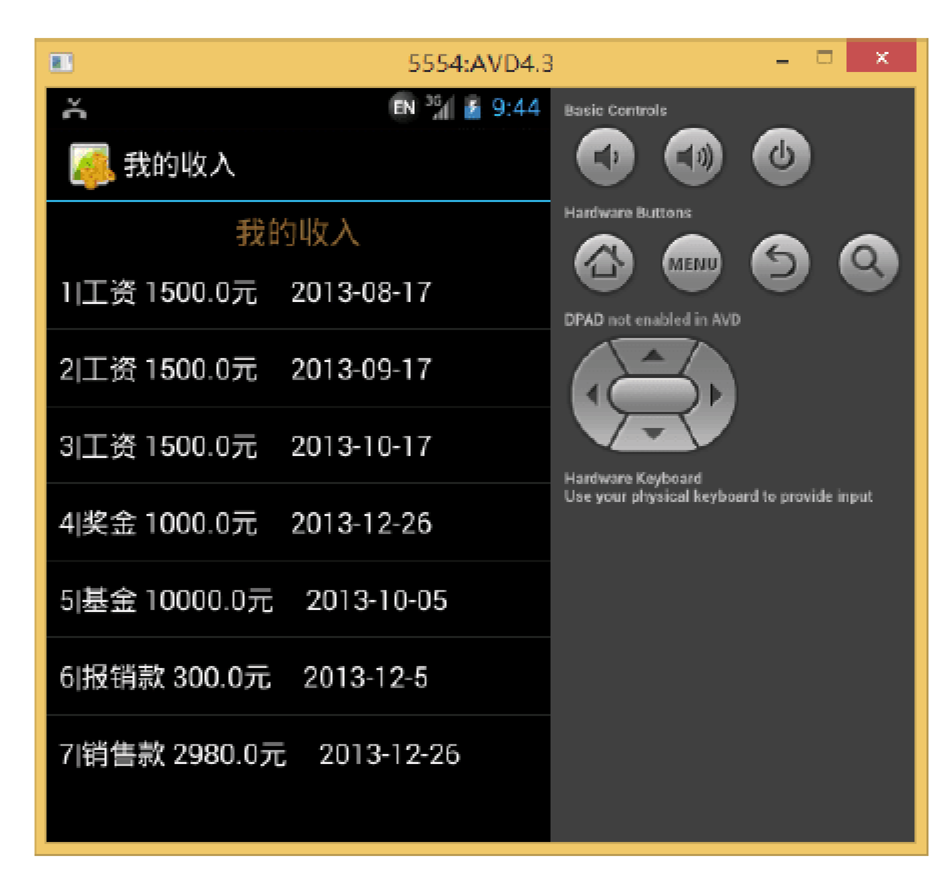
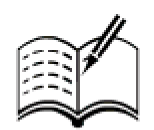


图 21.8 收入信息浏览窗体

在 res\layout 目录下新建一个 inaccountinfo.xml 文件，用来作为收入信息浏览窗体的布局文



件，该布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加一个 TextView 组件和一个 ListView 组件。其代码如下：



Note

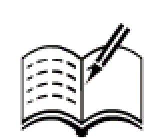
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/iteminfo" android:orientation="vertical"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:weightSum="1">
    <LinearLayout android:id="@+id/linearLayout1"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:orientation="vertical"
            android:layout_weight="0.06">
        <RelativeLayout android:layout_height="wrap_content"
            android:layout_width="match_parent">
        <TextView android:text="我的收入"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:textSize="20dp"
            android:textColor="#8C6931"
        />
        </RelativeLayout>
        </LinearLayout>
        <LinearLayout android:id="@+id/linearLayout2"
            android:layout_height="wrap_content"
            android:layout_width="match_parent"
            android:orientation="vertical"
            android:layout_weight="0.94">
        <ListView android:id="@+id/lvinaccountinfo"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scrollbarAlwaysDrawVerticalTrack="true"
            />
    </LinearLayout>
</LinearLayout>
```

21.9.6　显示所有的收入信息

在 com.xiaoke.accountsoft.activity 包中创建一个 Inaccountinfo.java 文件，该文件的布局文件设置为 inaccountinfo.xml。在 Inaccountinfo.java 文件中，首先创建类中需要用到的全局对象及变量。其代码如下：

```
public static final String FLAG = "id";                 //定义一个常量，用来作为请求码
ListView lvinfo;                                        //创建 ListView 对象
String strType = "";                                    //创建字符串，记录管理类型
```





在 onCreate()覆写方法中，初始化创建的 ListView 对象，并显示所有的收入信息。其代码如下：

```
lvinfo=(ListView) findViewById(R.id.lvinaccountinfo);          //获取布局文件中的 ListView 组件
ShowInfo(R.id.btnininfo);                                      //调用自定义方法显示收入信息
```

上面的代码中用到了 ShowInfo()方法，该方法用来根据参数中传入的管理类型 id 显示相应的信息。其代码如下：

```
private void ShowInfo(int intType) {                           //用来根据管理类型，显示相应的信息
    String[] strInfos = null;                                  //定义字符串数组，用来存储收入信息
    ArrayAdapter<String> arrayAdapter = null;                  //创建 ArrayAdapter 对象
    strType="btnininfo";                                       //为 strType 变量赋值
    InaccountDAO inaccountinfo=new InaccountDAO(Inaccountinfo.this);//创建 InaccountDAO
    //获取所有收入信息，并存储到 List 泛型集合中
    List<Tb_inaccount> listinfos=inaccountinfo.getScrollData(0, (int) inaccountinfo.getCount());
    strInfos=new String[listinfos.size()];                     //设置字符串数组的长度
    int m=0;                                                   //定义一个开始标识
    for (Tb_inaccount tb_inaccount:listinfos) {                //遍历 List 泛型集合
        //将收入相关信息组合成一个字符串，存储到字符串数组的相应位置
        strInfos[m]=tb_inaccount.getid()+"|"+tb_inaccount.getType()+"  "+String.valueOf(tb_inaccount.
getMoney())+"元     "+tb_inaccount.getTime();
        m++;                                                   //标识加 1
    }
    //使用字符串数组初始化 ArrayAdapter 对象
    arrayAdapter=new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, strInfos);
    lvinfo.setAdapter(arrayAdapter);                           //为 ListView 列表设置数据源
}
```

21.9.7　单击指定项时打开详细信息

当用户单击 ListView 列表中的某条收入记录时，为其设置监听事件，在监听事件中，根据用户单击的收入信息的编号，打开相应的 Activity。其代码如下：

```
lvinfo.setOnItemClickListener(new OnItemClickListener()                //为 ListView 添加项单击事件
{
    //覆写 onItemClick()方法
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        String strInfo=String.valueOf(((TextView) view).getText()); //记录收入信息
        String strid=strInfo.substring(0, strInfo.indexOf('|'));     //从收入信息中截取收入编号
        Intent intent = new Intent(Inaccountinfo.this, InfoManage.class);//创建 Intent
        intent.putExtra(FLAG, new String[]{strid,strType});           //设置传递数据
        startActivity(intent);                                       //执行 Intent 操作
    }
});
```





Note

21.9.8　设计修改/删除收入布局文件

修改/删除收入信息窗体的运行效果如图 21.9 所示。

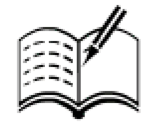


图 21.9　修改/删除收入信息

在 res\layout 目录下新建一个 infomanage.xml 文件，用来作为修改、删除收入信息和支出信息窗体的布局文件，该布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加 5 个 TextView 组件、4 个 EditText 组件、1 个 Spinner 组件和 2 个 Button 组件。其实现代码如下。

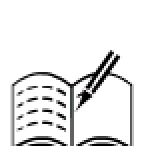
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/inoutitem"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
        <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="3"
            >
            <TextView android:id="@+id/inouttitle"
                android:layout_width="wrap_content"
                android:layout_gravity="center"
                android:gravity="center_horizontal"
                android:text="支出管理"
                android:textColor="#ffffff"
                android:textSize="40sp"
                android:textStyle="bold"
```





```
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="10dp"
            >
            <TextView android:layout_width="90dp"
            android:id="@+id/tvInOutMoney"
            android:textSize="20sp"
            android:text="金　额："
            android:layout_height="wrap_content"
            android:layout_alignBaseline="@+id/txtInOutMoney"
            android:layout_alignBottom="@+id/txtInOutMoney"
            android:layout_alignParentLeft="true"
            android:layout_marginLeft="16dp">
            </TextView>
            <EditText
            android:id="@+id/txtInOutMoney"
            android:layout_width="210dp"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/tvInOutMoney"
            android:inputType="number"
            android:numeric="integer"
            android:maxLength="9"
            />
            <TextView android:layout_width="90dp"
            android:id="@+id/tvInOutTime"
            android:textSize="20sp"
            android:text="时　间："
            android:layout_height="wrap_content"
            android:layout_alignBaseline="@+id/txtInOutTime"
            android:layout_alignBottom="@+id/txtInOutTime"
            android:layout_toLeftOf="@+id/txtInOutMoney">
            </TextView>
            <EditText
            android:id="@+id/txtInOutTime"
            android:layout_width="210dp"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@id/tvInOutTime"
            android:layout_below="@id/txtInOutMoney"
            android:inputType="datetime"
            />
            <TextView android:layout_width="90dp"
            android:id="@+id/tvInOutType"
```

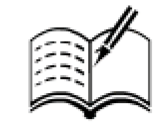




Note

```
android:textSize="20sp"
android:text="类　别："
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/spInOutType"
android:layout_alignBottom="@+id/spInOutType"
android:layout_alignLeft="@+id/tvInOutTime">
</TextView>
<Spinner android:id="@+id/spInOutType"
android:layout_width="210dp"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/tvInOutType"
android:layout_below="@id/txtInOutTime"
android:entries="@array/type"
android:textColor="#000000"
/>
<TextView android:layout_width="90dp"
android:id="@+id/tvInOut"
android:textSize="20sp"
android:text="付款方："
android:layout_height="wrap_content"
android:layout_alignBaseline="@+id/txtInOut"
android:layout_alignBottom="@+id/txtInOut"
android:layout_toLeftOf="@+id/spInOutType">
</TextView>
<EditText
android:id="@+id/txtInOut"
android:layout_width="210dp"
android:layout_height="wrap_content"
android:layout_toRightOf="@id/tvInOut"
android:layout_below="@id/spInOutType"
android:singleLine="false"
/>
<TextView android:layout_width="90dp"
android:id="@+id/tvInOutMark"
android:textSize="20sp"
android:text="备　注："
android:layout_height="wrap_content"
android:layout_alignTop="@+id/txtInOutMark"
android:layout_toLeftOf="@+id/txtInOut">
</TextView>
<EditText
android:id="@+id/txtInOutMark"
android:layout_width="210dp"
android:layout_height="150dp"
android:layout_toRightOf="@id/tvInOutMark"
android:layout_below="@id/txtInOut"
android:gravity="top"
android:singleLine="false"
/>
</RelativeLayout>
```






```
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="3"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="10dp"
            >
        <Button
            android:id="@+id/btnInOutDelete"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dp"
            android:text="删除"
            />
            <Button
             android:id="@+id/btnInOutEdit"
             android:layout_width="80dp"
            android:layout_height="wrap_content"
             android:layout_toLeftOf="@id/btnInOutDelete"
             android:text="修改"
            />
        </RelativeLayout>
    </LinearLayout>
</LinearLayout>
```

说明：

修改、删除收入信息和支出信息的布局文件都是使用 infomanage.xml 实现的。

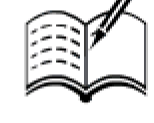
21.9.9 显示指定编号的收入信息

在 com.xiaoke.accountsoft.activity 包中创建一个 InfoManage.java 文件，该文件的布局文件设置为 infomanage.xml。在 InfoManage.java 文件中，首先创建类中需要用到的全局对象及变量。其代码如下：

```
protected static final int DATE_DIALOG_ID = 0;      //创建日期对话框常量
TextView tvtitle,textView;                          //创建 2 个 TextView 对象
EditText txtMoney,txtTime,txtHA,txtMark;            //创建 4 个 EditText 对象
Spinner spType;                                     //创建 Spinner 对象
Button btnEdit,btnDel;                              //创建 2 个 Button 对象
String[] strInfos;                                  //定义字符串数组
```




```
String strid,strType;                       //定义两个字符串变量，分别用来记录信息编号和管理类型
private int mYear;                          //年
private int mMonth;                         //月
private int mDay;                           //日
OutaccountDAO outaccountDAO=new OutaccountDAO(InfoManage.this);//创建 OutaccountDAO 对象
InaccountDAO inaccountDAO=new InaccountDAO(InfoManage.this);     //创建 InaccountDAO 对象
```



Note

说明：

修改、删除收入信息和支出信息的功能都是在 InfoManage.java 文件中实现的，所以在 21.9.10 节和 21.9.11 节中讲解修改、删除收入信息时，可能会涉及支出信息的修改与删除。

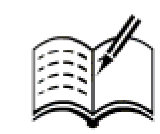
在 onCreate()覆写方法中，初始化创建的 EidtText、Spinner 对象和 Button 对象。其代码如下：

```
tvtitle=(TextView) findViewById(R.id.inouttitle);            //获取标题标签对象
textView=(TextView) findViewById(R.id.tvInOut);              //获取地点/付款方标签对象
txtMoney=(EditText) findViewById(R.id.txtInOutMoney);        //获取“金额”文本框
txtTime=(EditText) findViewById(R.id.txtInOutTime);          //获取“时间”文本框
spType=(Spinner) findViewById(R.id.spInOutType);             //获取类别下拉列表
txtHA=(EditText) findViewById(R.id.txtInOut);                //获取“地点/付款方”文本框
txtMark=(EditText) findViewById(R.id.txtInOutMark);          //获取“备注”文本框
btnEdit=(Button) findViewById(R.id.btnInOutEdit);            //获取“修改”按钮
btnDel=(Button) findViewById(R.id.btnInOutDelete);           //获取“删除”按钮
```

在 onCreate()覆写方法中初始化各组件对象后，使用字符串记录传入的 id 和类型，并根据类型判断显示收入信息还是支出信息。其代码如下：

```
Intent intent=getIntent();                                   //创建 Intent 对象
Bundle bundle=intent.getExtras();                            //获取传入的数据，并使用 Bundle 记录
strInfos=bundle.getStringArray(Showinfo.FLAG);               //获取 Bundle 中记录的信息
strid=strInfos[0];                                           //记录 id
strType=strInfos[1];                                         //记录类型
if(strType.equals("btnoutinfo"))                             //如果类型是 btnoutinfo
{
    tvtitle.setText("支出管理");                             //设置标题为“支出管理”
    textView.setText("地　点：");                            //设置“地点/付款方”标签文本为“地　点：”
    //根据编号查找支出信息，并存储到 Tb_outaccount 对象中
    Tb_outaccount tb_outaccount=outaccountDAO.find(Integer.parseInt(strid));
    txtMoney.setText(String.valueOf(tb_outaccount.getMoney()));   //显示金额
    txtTime.setText(tb_outaccount.getTime());                //显示时间
    spType.setPrompt(tb_outaccount.getType());               //显示类别
    txtHA.setText(tb_outaccount.getAddress());               //显示地点
    txtMark.setText(tb_outaccount.getMark());                //显示备注
}
else if(strType.equals("btnininfo"))                         //如果类型是 btnininfo
{
    tvtitle.setText("收入管理");                             //设置标题为“收入管理”
    textView.setText("付款方：");                            //设置“地点/付款方”标签文本为“付款方：”
    //根据编号查找收入信息，并存储到 Tb_outaccount 对象中
```






```
        Tb_inaccount tb_inaccount= inaccountDAO.find(Integer.parseInt(strid));
        txtMoney.setText(String.valueOf(tb_inaccount.getMoney()));          //显示金额
        txtTime.setText(tb_inaccount.getTime());                            //显示时间
        spType.setPrompt(tb_inaccount.getType());                           //显示类别
        txtHA.setText(tb_inaccount.getHandler());                           //显示付款方
        txtMark.setText(tb_inaccount.getMark());                            //显示备注
}
```

21.9.10 修改收入信息

当用户修改完显示的收入或者支出信息后，单击“修改”按钮，如果显示的是支出信息，则调用 OutaccountDAO 对象的 update()方法修改支出信息；如果显示的是收入信息，则调用 InaccountDAO 对象的 update()方法修改收入信息。其代码如下：

```
btnEdit.setOnClickListener(new OnClickListener() {                           //为“修改”按钮设置监听事件
        @Override
        public void onClick(View arg0) {
            //TODO Auto-generated method stub
            if(strType.equals("btnoutinfo"))                                //判断类型如果是 btnoutinfo
            {
                Tb_outaccount tb_outaccount=new Tb_outaccount();            //创建 Tb_outaccount 对象
                tb_outaccount.setid(Integer.parseInt(strid));               //设置编号
                //设置金额
                tb_outaccount.setMoney(Double.parseDouble(txtMoney.getText().toString()));
                tb_outaccount.setTime(txtTime.getText().toString());        //设置时间
                tb_outaccount.setType(spType.getSelectedItem().toString());     //设置类别
                tb_outaccount.setAddress(txtHA.getText().toString());       //设置地点
                tb_outaccount.setMark(txtMark.getText().toString());        //设置备注
                 outaccountDAO.update(tb_outaccount);                       //更新支出信息
            }
            else if(strType.equals("btninInfo"))                            //判断类型如果是 btnininfo
            {
                Tb_inaccount tb_inaccount=new Tb_inaccount();               //创建 Tb_inaccount 对象
                tb_inaccount.setid(Integer.parseInt(strid));                //设置编号
                //设置金额
                tb_inaccount.setMoney(Double.parseDouble(txtMoney.getText().toString()));
                tb_inaccount.setTime(txtTime.getText().toString());         //设置时间
                tb_inaccount.setType(spType.getSelectedItem().toString());  //设置类别
                tb_inaccount.setHandler(txtHA.getText().toString());        //设置付款方
                tb_inaccount.setMark(txtMark.getText().toString());         //设置备注
                 inaccountDAO.update(tb_inaccount);                         //更新收入信息
            }
            //弹出信息提示
            Toast.makeText(InfoManage.this, "〖数据〗修改成功！", Toast.LENGTH_SHORT).show();
        }
});
```




Note

21.9.11 删除收入信息

单击“删除”按钮，如果显示的是支出信息，则调用 OutaccountDAO 对象的 detele()方法删除支出信息；如果显示的是收入信息，则调用 InaccountDAO 对象的 detele()方法删除收入信息。其代码如下：

```
btnDel.setOnClickListener(new OnClickListener() {                    //为“删除”按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        if(strType.equals("btnoutinfo"))                              //判断类型如果是 btnoutinfo
        {
            outaccountDAO.detele(Integer.parseInt(strid));           //根据编号删除支出信息
        }
        else if(strType.equals("btnininfo"))                          //判断类型如果是 btnininfo
        {
            inaccountDAO.detele(Integer.parseInt(strid));            //根据编号删除收入信息
        }
        Toast.makeText(InfoManage.this, "〖数据〗删除成功！", Toast.LENGTH_SHORT).show();
    }
});
```

21.10 便签管理模块设计

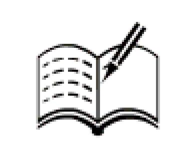
便签管理模块主要包括 3 部分，分别是“新增便签”、“便签信息浏览”和“修改/删除便签信息”，其中，“新增便签”用来添加便签信息；“便签信息浏览”用来显示所有的便签信息；“修改/删除便签信息”用来根据编号修改或者删除便签信息。本节将从这 3 个方面对便签管理模块进行详细介绍。

首先来看新增便签模块，新增便签窗体的运行效果如图 21.10 所示。



图 21.10　新增便签



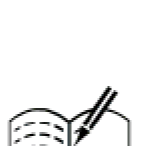


Note

21.10.1　设计新增便签布局文件

在 res\layout 目录下新建一个 accountflag.xml 文件，用来作为新增便签窗体的布局文件，该布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加两个 TextView 组件、一个 EditText 组件和两个 Button 组件。其实现代码如下：

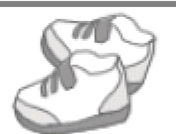
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/itemflag"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="3"
        >
        <TextView
            android:layout_width="wrap_content"
            android:layout_gravity="center"
            android:gravity="center_horizontal"
            android:text="新增便签"
            android:textSize="40sp"
            android:textColor="#ffffff"
            android:textStyle="bold"
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        >
        <RelativeLayout android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:padding="5dp"
           >
           <TextView android:layout_width="350dp"
           android:id="@+id/tvFlag"
           android:textSize="23sp"
           android:text="请输入便签，最多输入 200 字"
           android:textColor="#8C6931"
           android:layout_alignParentRight="true"
           android:layout_height="wrap_content"
           />
```

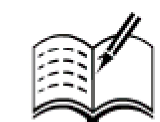

Note

```
                <EditText
                android:id="@+id/txtFlag"
                android:layout_width="350dp"
                android:layout_height="400dp"
                android:layout_below="@id/tvFlag"
                android:gravity="top"
                android:singleLine="false"
                />
                </RelativeLayout>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="3"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="10dp"
            >
        <Button
            android:id="@+id/btnflagCancel"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dp"
            android:text="取消"
            />
            <Button
             android:id="@+id/btnflagSave"
             android:layout_width="80dp"
            android:layout_height="wrap_content"
             android:layout_toLeftOf="@id/btnflagCancel"
             android:text="保存"
             android:maxLength="200"
            />
        </RelativeLayout>
    </LinearLayout>
</LinearLayout>
```

21.10.2 添加便签信息

在 com.xiaoke.accountsoft.activity 包中创建一个 Accountflag.java 文件，该文件的布局文件设置为 accountflag.xml。在 Accountflag.java 文件中，首先创建类中需要用到的全局对象及变量。其代码如下：






```
EditText txtFlag;                                              //创建 EditText 组件对象
Button btnflagSaveButton;                                      //创建 Button 组件对象
Button btnflagCancelButton;                                    //创建 Button 组件对象
```

在 onCreate()覆写方法中，初始化创建的 EidtText 和 Button 对象。其代码如下：

```
txtFlag=(EditText) findViewById(R.id.txtFlag);                 //获取便签文本框
btnflagSaveButton=(Button) findViewById(R.id.btnflagSave);     //获取"保存"按钮
btnflagCancelButton=(Button) findViewById(R.id.btnflagCancel); //获取"取消"按钮
```

填写完信息后，单击“保存”按钮，为该按钮设置监听事件，在监听事件中，使用 FlagDAO 对象的 add()方法将用户的输入保存到便签信息表中。其代码如下：

```
btnflagSaveButton.setOnClickListener(new OnClickListener() {   //为"保存"按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        String strFlag= txtFlag.getText().toString();          //获取便签文本框的值
        if(!strFlag.isEmpty()){                                //判断获取的值不为空
            FlagDAO flagDAO=new FlagDAO(Accountflag.this); //创建 FlagDAO 对象
            Tb_flag tb_flag=new Tb_flag(flagDAO.getMaxId()+1, strFlag);//创建 Tb_flag 对象
            flagDAO.add(tb_flag);                              //添加便签信息
            Toast.makeText(Accountflag.this, "〖新增便签〗数据添加成功！",Toast.LENGTH_SHORT).show();
        }
        else {
            Toast.makeText(Accountflag.this, "请输入便签！",Toast.LENGTH_SHORT).show();
        }
    }
});
```

21.10.3 清空便签文本框

单击“取消”按钮，清空便签文本框中的内容。其代码如下：

```
btnflagCancelButton.setOnClickListener(new OnClickListener() {  //为"取消"按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        txtFlag.setText("");                                   //清空"便签"文本框
    }
});
```

21.10.4 设计便签信息浏览布局文件

便签信息浏览窗体的运行效果如图 21.11 所示。




Note



图 21.11　便签信息浏览

说明：

便签信息浏览功能是在数据管理窗体中实现的，该窗体的布局文件是 showinfo.xml，对应的 java 文件是 Showinfo.java，所以下面讲解时，会通过对 showinfo.xml 布局文件和 Showinfo.java 文件的讲解，来介绍便签信息浏览功能的实现过程。

在 res\layout 目录下新建一个 showinfo.xml 文件，用来作为数据管理窗体的布局文件，该布局文件可以浏览支出信息、收入信息和便签信息。showinfo.xml 布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加 3 个 Button 组件和 1 个 ListView 组件。其代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/iteminfo" android:orientation="vertical"
     android:layout_width="wrap_content" android:layout_height="wrap_content"
     android:layout_marginTop="5dp"
     android:weightSum="1">
     <LinearLayout android:id="@+id/linearLayout1"
               android:layout_height="wrap_content"
               android:layout_width="match_parent"
               android:orientation="vertical"
               android:layout_weight="0.06">
          <RelativeLayout android:layout_height="wrap_content"
               android:layout_width="match_parent">
          <Button android:text="支出信息"
               android:id="@+id/btnoutinfo"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:textSize="20dp"
               android:textColor="#8C6931"
          />
```

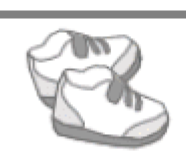


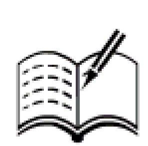

```
            <Button android:text="收入信息"
                android:id="@+id/btnininfo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_toRightOf="@id/btnoutinfo"
                android:textSize="20dp"
                android:textColor="#8C6931"
                />
            <Button android:text="便签信息"
                android:id="@+id/btnflaginfo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_toRightOf="@id/btnininfo"
                android:textSize="20dp"
                android:textColor="#8C6931"
                />
            </RelativeLayout>
            </LinearLayout>
            <LinearLayout android:id="@+id/linearLayout2"
                android:layout_height="wrap_content"
                android:layout_width="match_parent"
                android:orientation="vertical"
                android:layout_weight="0.94">
            <ListView android:id="@+id/lvinfo"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:scrollbarAlwaysDrawVerticalTrack="true"
                />
        </LinearLayout>
</LinearLayout>
```

21.10.5　显示所有的便签信息

在 com.xiaoke.accountsoft.activity 包中创建一个 Showinfo.java 文件，该文件的布局文件设置为 showinfo.xml。单击“便签信息”按钮，为该按钮设置监听事件，在监听事件中调用 ShowInfo() 方法显示便签信息。其代码如下：

```
btnflaginfo.setOnClickListener(new OnClickListener() {            //为“便签信息”按钮设置监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        ShowInfo(R.id.btnflaginfo);                               //显示便签信息
    }
});
```



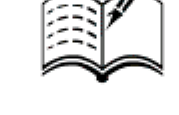


Note

上面的代码中用到了 ShowInfo()方法，该方法为自定义的无返回值类型方法，主要用来根据传入的管理类型显示相应的信息，该方法中有一个 int 类型的参数，用来表示传入的管理类型，该参数的取值主要包括 R.id.btnoutinfo、R.id.btnininfo 和 R.id.btnflaginfo 3 个值，分别用来显示支出信息、收入信息和便签信息。ShowInfo()方法的代码如下：

```
private void ShowInfo(int intType) {                    //用来根据传入的管理类型，显示相应的信息
    String[] strInfos = null;                           //定义字符串数组，用来存储收入信息
    ArrayAdapter<String> arrayAdapter = null;           //创建 ArrayAdapter 对象
    switch (intType) {                                  //以 intType 为条件进行判断
    case R.id.btnoutinfo:                               //如果是 btnoutinfo 按钮
        strType="btnoutinfo";                           //为 strType 变量赋值
        //创建 OutaccountDAO 对象
        OutaccountDAO outaccountinfo=new OutaccountDAO(Showinfo.this);
        //获取所有支出信息，并存储到 List 泛型集合中
        List<Tb_outaccount> listoutinfos=outaccountinfo.getScrollData(0, (int) outaccountinfo.getCount());
        strInfos=new String[listoutinfos.size()];       //设置字符串数组的长度
        int i=0;//定义一个开始标识
        for (Tb_outaccount tb_outaccount:listoutinfos) {//遍历 List 泛型集合
            //将支出相关信息组合成一个字符串，存储到字符串数组的相应位置
            strInfos[i]=tb_outaccount.getid()+"|"+tb_outaccount.getType()+"
"+String.valueOf(tb_outaccount.getMoney())+"元        "+tb_outaccount.getTime();
            i++;                                        //标识加 1
        }
        break;
    case R.id.btnininfo:                                //如果是 btnininfo 按钮
        strType="btnininfo";                            //为 strType 变量赋值
        //创建 InaccountDAO 对象
        InaccountDAO inaccountinfo=new InaccountDAO(Showinfo.this);
        //获取所有收入信息，并存储到 List 泛型集合中
        List<Tb_inaccount> listinfos=inaccountinfo.getScrollData(0, (int) inaccountinfo.getCount());
        strInfos=new String[listinfos.size()];          //设置字符串数组的长度
        int m=0;                                        //定义一个开始标识
        for (Tb_inaccount tb_inaccount:listinfos) { //遍历 List 泛型集合
            //将收入相关信息组合成一个字符串，存储到字符串数组的相应位置
            strInfos[m]=tb_inaccount.getid()+"|"+tb_inaccount.getType()+"
"+String.valueOf(tb_inaccount.getMoney())+"元        "+tb_inaccount.getTime();
            m++;                                        //标识加 1
        }
        break;
    case R.id.btnflaginfo:                              //如果是 btnflaginfo 按钮
        strType="btnflaginfo";                          //为 strType 变量赋值
        FlagDAO flaginfo=new FlagDAO(Showinfo.this);    //创建 FlagDAO 对象
        //获取所有便签信息，并存储到 List 泛型集合中
        List<Tb_flag> listFlags=flaginfo.getScrollData(0, (int) flaginfo.getCount());
        strInfos=new String[listFlags.size()];          //设置字符串数组的长度
```





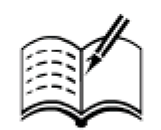

```
            int n=0;                                  //定义一个开始标识
            for (Tb_flag tb_flag:listFlags) {         //遍历 List 泛型集合
                //将便签相关信息组合成一个字符串，存储到字符串数组的相应位置
                strInfos[n]=tb_flag.getid()+"|"+tb_flag.getFlag();
                if(strInfos[n].length()>15)           //判断便签信息的长度是否大于 15
                //将位置大于 15 之后的字符串用……代替
                    strInfos[n]=strInfos[n].substring(0,15)+"……";
                n++;                                  //标识加 1
            }
            break;
        }
        //使用字符串数组初始化 ArrayAdapter 对象
        arrayAdapter=new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, strInfos);
        lvinfo.setAdapter(arrayAdapter);              //为 ListView 列表设置数据源
}
```

21.10.6　单击指定项时打开详细信息

当用户单击 ListView 列表中的某条便签记录时，为其设置监听事件，在监听事件中，根据用户单击的便签信息的编号，打开相应的 Activity。其代码如下：

```
lvinfo.setOnItemClickListener(new OnItemClickListener()                 //为 ListView 添加项单击事件
{
    //覆写 onItemClick()方法
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        String strInfo=String.valueOf(((TextView) view).getText()); //记录单击的项信息
        String strid=strInfo.substring(0, strInfo.indexOf('|'));    //从项信息中截取编号
        Intent intent = null;                                       //创建 Intent 对象
        if (strType=="btnoutinfo" | strType=="btnininfo") {          //判断如果是支出或者收入信息
            //使用 InfoManage 窗口初始化 Intent 对象
            intent=new Intent(Showinfo.this, InfoManage.class);
            intent.putExtra(FLAG, new String[]{strid,strType});     //设置要传递的数据
        }
        else if (strType=="btnflaginfo") {                          //判断如果是便签信息
            //使用 FlagManage 窗口初始化 Intent 对象
            intent=new Intent(Showinfo.this, FlagManage.class);
            intent.putExtra(FLAG, strid);                           //设置要传递的数据
        }
        startActivity(intent);                                      //执行 Intent，打开相应的 Activity
    }
});
```





Note

21.10.7　设计修改/删除便签布局文件

修改/删除便签信息窗体的运行效果如图 21.12 所示。

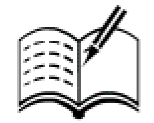


图 21.12　修改/删除便签信息

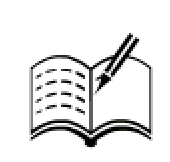
在 res\layout 目录下新建一个 flagmanage.xml 文件，用来作为修改、删除便签信息窗体的布局文件，该布局文件使用 LinearLayout 结合 RelativeLayout 进行布局，在该布局文件中添加两个 TextView 组件、一个 EditText 组件和两个 Button 组件。其实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/flagmanage"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
      <LinearLayout
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="3"
            >
            <TextView
                  android:layout_width="wrap_content"
                  android:layout_gravity="center"
                  android:gravity="center_horizontal"
                  android:text="便签管理"
                  android:textSize="40sp"
                  android:textColor="#ffffff"
                  android:textStyle="bold"
```





```
            android:layout_height="wrap_content"/>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp"
            >
            <TextView android:layout_width="350dp"
            android:id="@+id/tvFlagManage"
            android:textSize="23sp"
            android:text="请输入便签，最多输入 200 字"
            android:textColor="#8C6931"
            android:layout_alignParentRight="true"
            android:layout_height="wrap_content"
            />
            <EditText
            android:id="@+id/txtFlagManage"
            android:layout_width="350dp"
            android:layout_height="400dp"
            android:layout_below="@id/tvFlagManage"
            android:gravity="top"
            android:singleLine="false"
            />
            </RelativeLayout>
    </LinearLayout>
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="3"
        >
        <RelativeLayout android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="10dp"
            >
        <Button
            android:id="@+id/btnFlagManageDelete"
            android:layout_width="80dp"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dp"
            android:text="删除"
            />
            <Button
            android:id="@+id/btnFlagManageEdit"
```


Note

```
                android:layout_width="80dp"
                android:layout_height="wrap_content"
                android:layout_toLeftOf="@id/btnFlagManageDelete"
                android:text="修改"
                android:maxLength="200"
                />
            </RelativeLayout>
        </LinearLayout>
</LinearLayout>
```

21.10.8　显示指定编号的便签信息

在 com.xiaoke.accountsoft.activity 包中创建一个 FlagManage.java 文件，该文件的布局文件设置为 flagmanage.xml。在 FlagManage.java 文件中，首先创建类中需要用到的全局对象及变量。其代码如下：

```
EditText txtFlag;                                          //创建 EditText 对象
Button btnEdit,btnDel;                                     //创建两个 Button 对象
String strid;                                              //创建字符串，表示便签的 id
```

在 onCreate()覆写方法中，初始化创建的 EidtText 对象和 Button 对象。其代码如下：

```
txtFlag=(EditText) findViewById(R.id.txtFlagManage);       //获取便签文本框
btnEdit=(Button) findViewById(R.id.btnFlagManageEdit);     //获取“修改”按钮
btnDel=(Button) findViewById(R.id.btnFlagManageDelete);    //获取“删除”按钮
```

在 onCreate()覆写方法中初始化各组件对象后，使用字符串记录传入的 id，并根据该 id 显示便签信息。其代码如下：

```
Intent intent=getIntent();                                 //创建 Intent 对象
Bundle bundle=intent.getExtras();                          //获取便签 id
strid=bundle.getString(Showinfo.FLAG);                     //将便签 id 转换为字符串
final FlagDAO flagDAO=new FlagDAO(FlagManage.this);        //创建 FlagDAO 对象
//根据便签 id 查找便签信息，并显示在文本框中
txtFlag.setText(flagDAO.find(Integer.parseInt(strid)).getFlag());
```

21.10.9　修改便签信息

当用户修改完显示的便签信息后，单击“修改”按钮，调用 FlagDAO 对象的 update()方法即可完成指定修改。其代码如下：

```
btnEdit.setOnClickListener(new OnClickListener() {         //为“修改”按钮设置监听事件
    @Override
    public void onClick(View arg0) {
```






```
            //TODO Auto-generated method stub
            Tb_flag tb_flag=new Tb_flag();                    //创建 Tb_flag 对象
            tb_flag.setid(Integer.parseInt(strid));           //设置便签 id
            tb_flag.setFlag(txtFlag.getText().toString());    //设置便签值
            flagDAO.update(tb_flag);                          //修改便签信息
            Toast.makeText(FlagManage.this, "〖便签数据〗修改成功！", Toast.LENGTH_SHORT).show();
        }
});
```

21.10.10　删除便签信息

单击“删除”按钮，调用 FlagDAO 对象的 detele()方法删除便签信息，并弹出信息提示。其代码如下：

```
btnDel.setOnClickListener(new OnClickListener() {            //为“删除”按钮设置监听事件
        @Override
        public void onClick(View arg0) {
            //TODO Auto-generated method stub
            flagDAO.detele(Integer.parseInt(strid));          //根据指定的 id 删除便签信息
            Toast.makeText(FlagManage.this, "〖便签数据〗删除成功！", Toast.LENGTH_SHORT).show();
        }
});
```

21.11　系统设置模块设计

系统设置模块主要对家庭理财通中的登录密码进行设置，系统设置窗体的运行效果如图 21.13 所示。

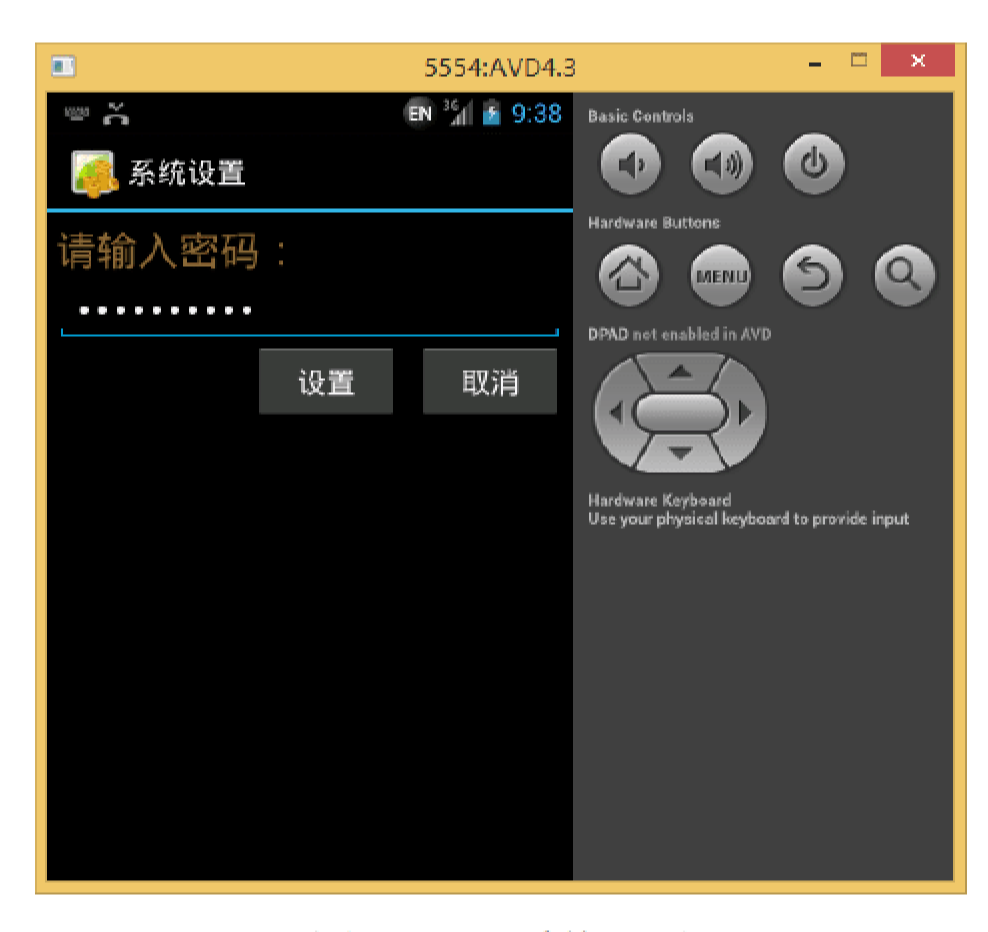
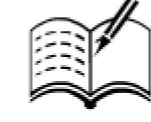


图 21.13　系统设置





说明：
在系统设置模块中，可以将登录密码设置为空。



Note

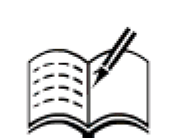
21.11.1 设计系统设置布局文件

在 res\layout 目录下新建一个 sysset.xml 文件，用来作为系统设置窗体的布局文件，该布局文件中，将布局方式修改为 RelativeLayout，然后添加一个 TextView 组件、一个 EditText 组件和两个 Button 组件。实现代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dp"
    >
    <TextView android:id="@+id/tvPwd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center_horizontal"
        android:text="请输入密码："
        android:textSize="25dp"
        android:textColor="#8C6931"
    />
    <EditText android:id="@+id/txtPwd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/tvPwd"
        android:inputType="textPassword"
        android:hint="请输入密码"
    />
    <Button android:id="@+id/btnsetCancel"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtPwd"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dp"
        android:text="取消"
    />
    <Button android:id="@+id/btnSet"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/txtPwd"
        android:layout_toLeftOf="@id/btnsetCancel"
        android:text="设置"
```



```
        />
</RelativeLayout>
```



Note

21.11.2 设置登录密码

在 com.xiaoke.accountsoft.activity 包中创建一个 Sysset.java 文件，该文件的布局文件设置为 sysset.xml。在 Sysset.java 文件中，首先创建一个 EidtText 对象和两个 Button 对象。其代码如下：

```
EditText txtpwd;                                              //创建 EditText 对象
Button btnSet,btnsetCancel;                                   //创建两个 Button 对象
```

在 onCreate()覆写方法中，初始化创建的 EidtText 对象和 Button 对象。其代码如下：

```
txtpwd=(EditText) findViewById(R.id.txtPwd);                  //获取密码文本框
btnSet=(Button) findViewById(R.id.btnSet);                    //获取“设置”按钮
btnsetCancel=(Button) findViewById(R.id.btnsetCancel);        //获取“取消”按钮
```

当用户单击“设置”按钮时，为“设置”按钮添加监听事件，在监听事件中，首先创建 PwdDAO 类的对象和 Tb_pwd 类的对象，然后判断数据库中是否已经设置密码，如果没有，则添加用户密码；否则，修改用户密码，最后弹出提示信息。其代码如下：

```
btnSet.setOnClickListener(new OnClickListener() {             //为“设置”按钮添加监听事件
    @Override
    public void onClick(View arg0) {
        //TODO Auto-generated method stub
        PwdDAO pwdDAO=new PwdDAO(Sysset.this);                 //创建 PwdDAO 对象
        Tb_pwd tb_pwd=new Tb_pwd(txtpwd.getText().toString()); //根据输入密码创建 Tb_pwd 对象
        if(pwdDAO.getCount()==0){                              //判断是否已经设置了密码
            pwdDAO.add(tb_pwd);                                //添加用户密码
        }
        else {
            pwdDAO.update(tb_pwd);                             //修改用户密码
        }
        //弹出信息提示
        Toast.makeText(Sysset.this, "〖密码〗设置成功！", Toast.LENGTH_SHORT).show();
    }
});
```

21.11.3 重置密码文本框

单击“取消”按钮，清空密码文本框，并为其设置初始提示。其代码如下：

```
btnsetCancel.setOnClickListener(new OnClickListener() {
    @Override
```

```
public void onClick(View arg0) {
    //TODO Auto-generated method stub
    txtpwd.setText("");                         //清空密码文本框
    txtpwd.setHint("请输入密码");                //为“密码”文本框设置提示
  }
});
```

21.12　开发常见问题与解决

21.12.1　程序在装有 Android 系统的手机上无法运行

问题描述：我有一款 HTC 的智能手机，为什么下载安装该程序后无法运行？

解决方法：该错误是由于 Android 版本低造成的，由于家庭理财通系统使用的是 Android 4.3 版本开发的，所以需要在装有 Android 4.3 以上版本的手机上运行，你可以联系供应商升级 Android 到最新版本，然后再安装使用。

21.12.2　无法将最新修改在 Android 模拟器中体现

问题描述：在 Eclipse 开发环境中修改完代码，重新运行程序时，出现如图 21.14 所示的错误提示。

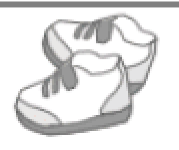
```
问题 | Javadoc | 声明 | LogCat | 控制台
Android
[2013-10-17 17:05:36 - AccountMS] Android Launch!
[2013-10-17 17:05:36 - AccountMS] adb is running normally.
[2013-10-17 17:05:36 - AccountMS] Performing com.xiaoke.accountsoft.activity.Login activity launch
[2013-10-17 17:05:36 - AccountMS] Automatic Target Mode: using existing emulator 'emulator-5554' running compatibl
[2013-10-17 17:05:37 - AccountMS] Uploading AccountMS.apk onto device 'emulator-5554'
[2013-10-17 17:05:38 - AccountMS] Failed to install AccountMS.apk on device 'emulator-5554': Connection refused:
[2013-10-17 17:05:38 - AccountMS] java.net.ConnectException: Connection refused: connect
[2013-10-17 17:05:38 - AccountMS] Launch canceled!
```

图 21.14　修改完代码再次运行时的错误提示

解决方法：这是由于 Android 使用超时引起的，Android 4.3 版的模拟器在使用一段时间后，会自动超时，从而导致有的修改无法在 Android 模拟器上体现，遇到这种情况，只需要关闭当前 Android 模拟器，并重新启动即可。

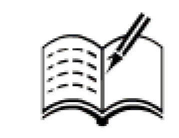
21.12.3　退出系统后还能使用记录的密码登录

问题描述：使用家庭理财通系统时，当用户单击 Android 模拟器的“返回”按钮，或者单击主窗体中的“退出”按钮时，返回“登录”窗口，这时“登录”窗口还记录着用户原来输入的密码，再次单击“登录”按钮，可以直接进入家庭理财通系统的主窗体。



解决方法：该问题主要是由于在登录时没有清空密码文本框造成的，解决该问题时，只需在“登录”按钮的监听事件中添加一段清空密码文本框的代码即可。其代码如下：

```
txtlogin.setText("");                                          //清空密码文本框
```



21.13 本章小结

本章重点讲解了家庭理财通系统中关键模块的开发过程、项目的运行及安装。通过对本章的学习，读者应该能够熟悉软件的开发流程，并重点掌握如何在 Android 项目中对多个不同的数据表进行添加、修改、删除以及查询等操作；另外，读者还应该掌握如何使用多种布局管理器对 Android 程序的界面进行布局。